Decidable Equality in a Logical Framework with Sigma Kinds

Robin Adams

November 3, 2004

1 Introduction

One of the main features of type theories that has made them so successful in the formalisation of mathematics is the fact that it is quite possible to have a decidable theory that is adequate for the formalisation of a large amount of mathematics. When implementing such a system, we do not need to find formal derivations of each of our theorems; we can forget about the axioms and primitive rules of deduction altogether, and just keep the decision procedure. There can be any number of methods of producing theorems, provided only that the decision algorithm agrees that each of them is derivable.

Almost all type theories have an equality relation, or convertibility relation, on their terms that plays a crucial role in the system, and proving this equality relation decidable is usually the main step in proving the theory decidable. In many theories, the equality relation is externally defined, usually with reference to some reduction or computation relation. Once it has been proven that two terms are equal iff they reduce to the same normal form, it is fairly trivial to prove the equality relation decidable. The hardest step in such a proof is always proving strong normalisation; that is, proving that any reduction sequence terminates in some normal form.

However, in many theories — notably Martin-Löf type theory and many systems intended as logical frameworks — the equality relation is a judgement form included in the system's rules. The objects of each type, and the types themselves, can have different criteria for equality. It is often a much harder prospect to prove this type of system decidable. If we wish to give the equality relation of such a system in terms of a reduction relation, that reduction relation needs to be dependent on the type of the term being reduced, and possibly also the context.

Luo's logical framework LF, described in [4], is one such system. Goguen [2] developed a system of *typed operational semantics* for LF that is essentially such a reduction relation. It is fairly easy to produce a proof of the decidability of LF from his work.

An alternative approach is possible. In 1991, Coquand [1] proved the decidability of equality in a system he simply called Type Theory — a type theory

with dependent products, $\beta\eta$-conversion, and a single universe à la Russell, closed predicatively under the formation of Π-types. His method involves giving directly the algorithm for deciding equality of weak-head normal forms, based on the structure of the normal forms. It is easy to see that each term has a unique weak-head normal form; two terms are equal in the system iff the algorithm decides that their weak-head normal forms are equal.

One immediate advantage of this method is that the algorithm is more efficient than reducing and comparing normal forms. In particular, when two terms are not equal, Coquand's algorithm will often show that they are not in a far shorter time than it would take to reduce both to normal form.

In 2000, Harper and Pfening [3] managed to simplify Coquand's method considerably. Coquand's algorithm takes any two terms, and decides whether there exists a type of which they are equal objects (or whether they are equal types). However, in the final proof of the decidability of the system, whenever we use the decidability of equality, we already know a type of which the two terms are both objects (or that the two terms are both types). We can allow the algorithm to make use of the information that the type provides.

Harper and Pfening give an algorithm that takes two terms a, b and a type A, and, given that $a : A$ and $b : A$, decides whether $a = b : A$. They use this method to prove the Edinburgh LF decidable. Their algorithm is not particularly more simple than Coquand's. However, Harper and Pfening's subsequent proof that the algorithm is correct and complete is notably simpler than Coquand's.

Harper and Pfening's method is also more adaptable than Coquand's. Coquand's method would be very difficult to apply to a system that had a greater variety of types than just Π-types. Harper and Pfening claim

> While it is beyond the scope of this paper, we believe that our construction is robust with respect to extension of the type theory with products, unit, linearity, subtyping and similar complicating factors.

This paper is, in part, a vindication of that claim.

I was asked by Randy Pollack to investigate the application of each of these methods to the question of whether a logical framework with Σ-kinds and a unit kind is decidable. I chose to work with an extension of Luo's LF. Because of the presence of the unit kind, an approach based on untyped reduction is probably impossible. Coquand's method failed for the same reason. Goguen's method quickly became unworkably complex; but, as can be seen in this paper, Harper and Pfening's method succeeded with very little adaptation.

1.1 Outline

This paper contains the application of Harper and Pfening's method to two systems, which I have named $LF + \Sigma$ and $LF + \Sigma_s$. Each is an extension of Luo's LF to include Σ-kinds and a unit kind. In $LF + \Sigma_s$, the pairs in the Σ-kinds are strongly typed; in $LF + \Sigma$, they are not.

It is a further testament to the adaptability of Harper and Pfening's method that the development for each system is almost identical. Throughout this paper, we work with $LF + \Sigma_s$, indicating the places at which the development for $LF + \Sigma$ requires more than simply removing the subscripts on the pairs.

I have omitted the proofs of almost all of the lemmata, as they are perfectly routine; and I have described the structure of those proofs that are not. I have given the proofs of the main theorems in full detail.

In Section 2, we give formal definitions of the two systems $LF + \Sigma$ and $LF + \Sigma_s$.

In Section 3, we give the basic metatheoretic properties of the systems that will be needed.

The plan for the rest of the paper is then as follows. We describe here only the development to show that equality between terms is decidable; a similar development for equality between kinds runs alongside it in the main body of the paper.

We build a simple (non-dependent) type theory with the same terms as $LF+\Sigma_s$, but whose only type constructors are $\times$ and $\to$. We define an operation of *erasure* (of dependencies), that maps a kind A of $LF + \Sigma_s$ to a kind A^- of the simple type theory.

We define an equality relation in the simple type theory, which we call *type directed term equality*,

$$\Delta \vdash a \Leftrightarrow b : \kappa,$$

between simple contexts Δ, simple kinds κ, and terms a and b. We shall show that this relation is equivalent to equality in $LF + \Sigma_s$; that is, if $\Gamma \vdash a : A$ and $\Gamma \vdash b : A$, then

$$\Gamma \vdash a = b : A \Leftrightarrow \Gamma^- \vdash a \Leftrightarrow b : A^- \tag{1}$$

Then, to show equality is decidable, it is sufficient to show that the type-directed equality relation is decidable.

In Section 4, we define the type-directed equality relation, and derive a few of its properties. We give the algorithm which — we claim — is a decision procedure for this relation. We show that, when the algorithm terminates, it does decide whether $\Delta \vdash a \Leftrightarrow b : \kappa$ correctly.

In Section 5, we prove the *Soundness Theorem*:

> If $\Gamma \vdash a = b : A$ then $\Gamma^- \vdash a \Leftrightarrow b : A^-$.

In order to do so, we introduce a halfway stage, the *Kripke logical relation*

$$\Delta \vdash a = b \in [\![\kappa]\!]$$

between simple contexts Δ, simple kinds κ and terms a, b. This is the sort of relation that one is used to seeing in the metatheory of a type theory; in effect, it gives a semantics for the theory built out of the terms of the theory itself. It is called a Kripke logical relation because it can be seen as using the simple contexts as possible worlds in a Kripke forcing semantics. In particular, the clause for $\to$-types,

$\Delta \vdash f = g \in [\![\kappa_1 \to \kappa_2]\!]$ iff, whenever $\Delta' \supseteq \Delta$ and $\Delta' \vdash a = b \in [\![\kappa_1]\!]$, then $\Delta' \vdash f(a) = g(b) \in [\![\kappa_2]\!]$

could have been written

$\Delta \vdash f = g \in [\![\kappa_1 \to \kappa_2]\!]$ iff $\Delta \Vdash \forall a, b(a = b \in [\![\kappa_1]\!] \Rightarrow f(a) = g(b) \in [\![\kappa_2]\!])$

The Kripke logical relation shall turn out to be equivalent both to the type-directed equality and the equality of $LF + \Sigma_s$; however, for the Soundness Theorem, all we need are the results

If $\Delta \vdash a = b \in [\![\kappa]\!]$ then $\Delta \vdash a \Leftrightarrow b : \kappa$.

If $\Gamma \vdash a = b : A$ then $\Gamma^- \vdash a = b \in [\![A^-]\!]$.

In Section 6, we prove the *Completeness Theorem*:

If $\Gamma \vdash a : A$, $\Gamma \vdash b : A$ and $\Gamma^- \vdash a \Leftrightarrow b : A^-$, then $\Gamma \vdash a = b : A$.

Together, the Soundness and Completeness theorems prove (1) above.

In Section 7, we prove that the algorithm presented in Section 4 always terminates when given well-typed terms, and we pull all our results together to show that equality in both theories is decidable. We also give an algorithm for type-checking in $LF+\Sigma_s$, and so show that $LF+\Sigma_s$ in its entirety is a decidable theory. I confess that I do not know whether type-checking is decidable in $LF + \Sigma$.

Finally, in Section 8, we show how the work we have done can be used to prove that $LF+\Sigma_s$ has the property of strengthening, which is usually the most difficult metatheoretic property to prove of a theory.

2 The Logical Frameworks $LF + \Sigma$ and $LF + \Sigma_s$

We define here the formal systems $LF + \Sigma$ and $LF + \Sigma_s$.

We will ostensibly be working with $LF+\Sigma_s$ throughout this paper; however, except for Lemma 3.15, the second half of section 7, and section 8, simply removing the subscripts on the pairs will give a development for $LF + \Sigma$.

The classes of *terms* and *kinds* of the language of $LF + \Sigma_s$ are defined by the following grammar:

$$\begin{array}{lcl} \text{Term } a & ::= & x \mid \lambda x : A.a \mid a(a) \mid (a, a)_{xA} \mid \pi_1(a) \mid \pi_2(a) \mid * \\ \text{Kind } A & ::= & \Pi x : A.A \mid \Sigma x : A.A \mid 1 \mid \mathit{Type} \mid \mathit{El}(a) \end{array}$$

The occurrences of x in a are bound in $\lambda x : A.a$; those in B are bound in $\Pi x : A.B$, $\Sigma x : A.B$ and $(a, b)_{xB}$. We identify terms up to α-conversion.

Denote by $FV(a)$ and $FV(A)$ the free variables occurring in the term a and the kind A, respectively.

Definition 2.1 (Context) *A* context *is a finite sequence of pairs $x : A$, where x is a variable and A a kind. We will denote the empty context by $\langle\rangle$.*

The context $x_1 : A_1, \ldots, x_n : A_n$ is consistent *iff each of $x_1, \ldots, x_n$ is distinct, and, for $i = 1, \ldots, n$,*

$$FV(A_i) \subseteq \{x_1, \ldots, x_{i-1}\}$$

Define

$$\begin{aligned} \text{dom}\ (x_1 : A_1, \ldots, x_n : A_n) &= \{x_1, \ldots, x_n\} \\ FV(x_1 : A_1, \ldots, x_n : A_n) &= \{x_1, \ldots, x_n\} \cup \bigcup_{i=1}^{n} FV(A_i) \end{aligned}$$

(For a consistent context, these will be equal.)

We write $\Gamma \subseteq \Delta$ (Δ extends *Γ), iff every element $x : A$ of Γ occurs in Δ.*

Definition 2.2 (Judgement) *A* judgement *is an expression of one of the following forms:*

- Γ valid
- $\Gamma \vdash a : A$
- $\Gamma \vdash A$ kind
- $\Gamma \vdash a = b : A$
- $\Gamma \vdash A = B$

where Γ is a context, a and b are terms and A and B are kinds.

We now present the rules of derivation of $LF + \Sigma_s$ (Figures 1 and 2).

The rules for $LF + \Sigma$ are found by removing the subscripts from the pairs, except that $(pair{-}eq)$ should become

$$(pair{-}eq)\frac{\Gamma \vdash a_1 = a_2 : A \quad \Gamma \vdash b_1 = b_2 : [a_1/x]B \quad \Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash (a_1, b_1) = (a_2, b_2) : \Sigma x : A.B}$$

3 Basic Metatheory

Before we present the algorithmic equality relation, we develop some metatheoretic results that we will need.

Lemma 3.1 (Context Validity) *Every derivation of $\Gamma, \Gamma' \vdash J$ has a subderivation of Γ* valid.

Lemma 3.2 (Weakening) *If $\Gamma \vdash J$, $\Gamma \subseteq \Delta$, and Δ* valid, *then $\Delta \vdash J$.*

Lemma 3.3 (Contexts) *If $\Gamma \vdash J$, then Γ is consistent, and $FV(J) \subseteq$ dom Γ.*

Contexts

$$(Emp)\ \frac{}{\langle\rangle\ \text{valid}} \qquad (Weak)\ \frac{\Gamma \vdash A\ \text{kind}}{\Gamma, x : A\ \text{valid}}\ (x \notin \text{dom}\ \Gamma) \qquad (Var)\ \frac{\Gamma\ \text{valid}}{\Gamma \vdash x : A}\ (x : A \in \Gamma)$$

Equality

$$(KRefl)\ \frac{\Gamma \vdash A\ \text{kind}}{\Gamma \vdash A = A} \qquad (KSym)\ \frac{\Gamma \vdash A = B}{\Gamma \vdash B = A} \qquad (KTrans)\ \frac{\Gamma \vdash A = B \quad \Gamma \vdash B = C}{\Gamma \vdash A = C}$$

$$(Refl)\ \frac{\Gamma \vdash a : A}{\Gamma \vdash a = a : A} \qquad (Sym)\ \frac{\Gamma \vdash a = b : A}{\Gamma \vdash b = a : A} \qquad (Trans)\ \frac{\Gamma \vdash a = b : A \quad \Gamma \vdash b = c : A}{\Gamma \vdash a = c : A}$$

$$(Eq)\ \frac{\Gamma \vdash a : A \quad \Gamma \vdash A = B}{\Gamma \vdash a : B} \qquad (=R)\ \frac{\Gamma \vdash a = b : A \quad \Gamma \vdash A = B}{\Gamma \vdash a = b : B}$$

Dependent Product Kinds

$$(\Pi)\ \frac{\Gamma, x : A \vdash B\ \text{kind}}{\Gamma \vdash \Pi x : A.B} \qquad (\Pi{-}eq)\ \frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash B_1 = B_2}{\Gamma \vdash \Pi x : A_1.B_1 = \Pi x : A_2.B_2}$$

$$(\lambda)\ \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x : A.b : \Pi x : A.B} \qquad (\lambda{-}eq)\ \frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash b_1 = b_2 : B}{\Gamma \vdash \lambda x : A_1.b_1 = \lambda x : A_2 : b_2 : \Pi x : A_1.B}$$

$$(app)\ \frac{\Gamma \vdash f : \Pi x : A.B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : [a/x]B} \qquad (app{-}eq)\ \frac{\Gamma \vdash f = g : \Pi x : A.B \quad \Gamma \vdash a = b : A}{\Gamma \vdash f(a) = g(b) : [a/x]B}$$

$$(\beta)\ \frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x : A.b)(a) = [a/x]b : [a/x]B} \qquad (\eta_\Pi)\ \frac{\Gamma \vdash f : \Pi x : A.B}{\Gamma \vdash \lambda x : A.f(x) = f : \Pi x : A.B}\ (x \notin FV(f))$$

Figure 1: The system $LF + \Sigma_s$

Dependent Sum Kinds

$$(\Sigma)\ \frac{\Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash \Sigma x : A.B} \qquad (\Sigma{-}eq)\ \frac{\Gamma \vdash A_1 = A_2 \quad \Gamma, x : A_1 \vdash B_1 = B_2}{\Gamma \vdash \Sigma x : A_1.B_1 = \Sigma x : A_2.B_2}$$

$$(pair)\ \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash (a, b)_{xB} : \Sigma x : A.B}$$

$$(pair{-}eq)\ \frac{\Gamma \vdash a_1 = a_2 : A \quad \Gamma \vdash b_1 = b_2 : [a_1/x]B_1 \quad \Gamma, x : A \vdash B_1 = B_2}{\Gamma \vdash (a_1, b_1)_{xB_1} = (a_2, b_2)_{xB_2} : \Sigma x : A.B_1}$$

$$(\pi_1)\ \frac{\Gamma \vdash a : \Sigma x : A.B}{\Gamma \vdash \pi_1(a) : A} \qquad (\pi_1{-}eq)\ \frac{\Gamma \vdash a = b : \Sigma x : A.B}{\Gamma \vdash \pi_1(a) = \pi_1(b) : A}$$

$$(\pi_2)\ \frac{\Gamma \vdash a : \Sigma x : A.B}{\Gamma \vdash \pi_2(a) : [\pi_1(a)/x]B} \qquad (\pi_2{-}eq)\ \frac{\Gamma \vdash a = b : \Sigma x : A.B}{\Gamma \vdash \pi_2(a) = \pi_2(b) : [\pi_1(a)/x]B}$$

$$(\sigma_1)\ \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash \pi_1((a, b)_{xB}) = a : A}$$

$$(\sigma_2)\ \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash \pi_2((a, b)_{xB}) = b : [a/x]B}$$

$$(\eta_\Sigma)\ \frac{\Gamma \vdash a : \Sigma x : A.B}{\Gamma \vdash (\pi_1(a), \pi_2(a))_{xB} = a : \Sigma x : A.B}$$

The Unit Kind

$$(1)\ \frac{\Gamma \text{ valid}}{\Gamma \vdash 1 \text{ kind}} \qquad (*)\ \frac{\Gamma \text{ valid}}{\Gamma \vdash * : 1} \qquad (\eta_1)\ \frac{\Gamma \vdash a : 1}{\Gamma \vdash * = a : 1}$$

The Kind of Types

$$(Type)\ \frac{\Gamma \text{ valid}}{\Gamma \vdash Type \text{ kind}} \qquad (El)\ \frac{\Gamma \vdash t : Type}{\Gamma \vdash El(t) \text{ kind}} \qquad (El{-}eq)\ \frac{\Gamma \vdash t = u : Type}{\Gamma \vdash El(t) = El(u)}$$

Figure 2: The system $LF + \Sigma_s$

Lemma 3.4 (Substitution) *If* $\Gamma \vdash a : A$ *and* $\Gamma, x : A, \Gamma' \vdash J$*, then* $\Gamma, [a/x]\Gamma' \vdash [a/x]J$.

Lemma 3.5 (Context Conversion) *If* $\Gamma \vdash B$ kind, $\Gamma, x : A \vdash J$ *and* $\Gamma \vdash B = A$*, then* $\Gamma, x : B \vdash J$.

Lemma 3.6 (Functionality for Typing) *1. If* $\Gamma \vdash a = b : A$, $\Gamma \vdash a : A$, $\Gamma \vdash b : A$, *and* $\Gamma, x : A, \Gamma' \vdash c : B$, *then* $\Gamma, [a/x]\Gamma' \vdash [a/x]c = [b/x]c : [a/x]B$.

2. If $\Gamma \vdash a = b : A$, $\Gamma \vdash a : A$, $\Gamma \vdash b : A$, *and* $\Gamma, x : A, \Gamma' \vdash B$ kind, *then* $\Gamma, [a/x]\Gamma' \vdash [a/x]B = [a/x]B'$.

Lemma 3.7 (Validity) *1. If* $\Gamma \vdash a : A$ *then* $\Gamma \vdash A$ kind.

2. If $\Gamma \vdash a = b : A$ *then* $\Gamma \vdash a : A$, $\Gamma \vdash b : A$ *and* $\Gamma \vdash A$ kind.

3. If $\Gamma \vdash A = B$ *then* $\Gamma \vdash A$ kind *and* $\Gamma \vdash B$ kind.

Lemma 3.8 (Functionality for Term Equality) *If* $\Gamma \vdash a = b : A$ *and* $\Gamma, x : A \vdash c = d : B$ *then* $\Gamma \vdash [a/x]c = [b/x]d : [a/x]B$.

Lemma 3.9 (Functionality for Kind Equality) *If* $\Gamma, x : A \vdash B = C$ *and* $\Gamma \vdash a = b : A$ *then* $\Gamma \vdash [a/x]B = [b/x]C$.

3.1 Typing Inversion

Lemma 3.10 (Typing Inversion for Variables) *If* $\Gamma \vdash x : A$ *then there exists* B *such that* $x : B \in \Gamma$ *and* $\Gamma \vdash A = B$.

Lemma 3.11 (Typing Inversion for λ-terms) *If* $\Gamma \vdash \lambda x : A.a : B$*, then there exists* A' *such that* $\Gamma \vdash B = \Pi x : A.A'$ *and* $\Gamma, x : A \vdash a : A'$.

Lemma 3.12 (Typing Inversion for Applications) *If* $\Gamma \vdash f(a) : A$*, then there exist* A_1, A_2 *such that* $\Gamma \vdash f : \Pi x : A_1.A_2$, $\Gamma \vdash a : A_1$ *and* $\Gamma \vdash [a/x]A_2 = A$.

Lemma 3.13 (Typing Inversion for π_1-terms) *If* $\Gamma \vdash \pi_1(a) : A$*, then there exists* B *such that* $\Gamma \vdash a : \Sigma x : A.B$.

Lemma 3.14 (Typing Inversion for π_2-terms) *If* $\Gamma \vdash \pi_2(a) : A$*, then there exist* B, C *such that* $\Gamma \vdash a : \Sigma x : B.C$ *and* $\Gamma \vdash [\pi_1(a)/x]C = A$.

Lemma 3.15 (Typing Inversion for Pairs) $(LF + \Sigma_s)$ *If* $\Gamma \vdash (a, b)_{xB} : A$*, then there exists* C *such that* $\Gamma \vdash a : C$, $\Gamma \vdash b : [a/x]B$, *and* $\Gamma \vdash \Sigma x : C.B = A$.

Lemma 3.16 (Typing Inversion for Pairs) $(LF + \Sigma)$ *If* $\Gamma \vdash (a, b) : A$*, then there exist* B, C *such that* $\Gamma \vdash a : C$, $\Gamma \vdash b : [a/x]B$, *and* $\Gamma \vdash \Sigma x : C.B = A$.

Lemma 3.17 (Typing Inversion for $*$) *If* $\Gamma \vdash * : A$ *then* $\Gamma \vdash 1 = A$.

3.2 Equality Inversion

Lemma 3.18 (Equality Inversion for $Type$) *If $\Gamma \vdash A = Type$ or $\Gamma \vdash Type = A$, then $A \equiv Type$.*

Lemma 3.19 (Equality Inversion for Π and Σ) *Let Q be Π or Σ. If $\Gamma \vdash A = Qx : B.C$ or $\Gamma \vdash Qx : B.C = A$, then there exist B', C' such that $A \equiv Qx : B'.C'$, $\Gamma \vdash B = B'$ and $\Gamma, x : B \vdash C = C'$.*

Lemma 3.20 (Equality Inversion for 1) *If $\Gamma \vdash A = 1$ or $\Gamma \vdash 1 = A$, then $A \equiv 1$.*

Lemma 3.21 (Equality Inversion for El) *If $\Gamma \vdash A = El(t)$ or $\Gamma \vdash El(t) = A$, then there exists u such that $A \equiv El(u)$ and $\Gamma \vdash t = u : Type$.*

3.3 Weak Head Reduction

Definition 3.22 (Weak Head Reduction) (One-step) Weak head reduction, *$\rhd_{whd}$, is the binary relation on terms defined recursively by*

$$(\beta)\ \frac{}{(\lambda x : A.a)(b) \rhd_{whd} [b/x]a} \qquad (app)\ \frac{f \rhd_{whd} g}{f(a) \rhd_{whd} g(a)}$$

$$(\sigma_1)\ \frac{}{\pi_1((a,b)_{xB}) \rhd_{whd} a} \qquad (\sigma_2)\ \frac{}{\pi_2((a,b)_{xB}) \rhd_{whd} b}$$

$$(\pi_1)\ \frac{a \rhd_{whd} b}{\pi_1(a) \rhd_{whd} \pi_1(b)} \qquad (\pi_2)\ \frac{a \rhd_{whd} b}{\pi_2(a) \rhd_{whd} \pi_2(b)}$$

A term a is a weak head normal form (whnf) *iff there does not exist b such that $a \rhd_{whd} b$.*

Lemma 3.23 (Determinacy of Weak Head Reduction) *If $a \rhd_{whd} b$ and $a \rhd_{whd} c$, then $b \equiv c$.*

4 Algorithmic Equality

We construct a simple type theory, and an operation of *erasure* that sends a kind of $LF + \Sigma_s$ to a kind of the simple type theory:

Definition 4.1 (Simple Kind, Simple Context) *The class of* simple kinds *is defined by the following grammar:*

$$\textit{Simple kind } \kappa ::= \kappa \times \kappa \mid \kappa \to \kappa \mid 1^- \mid Type^- \mid El^-$$

A simple context *is a finite sequence of pairs $x_1 : \kappa_1, \ldots, x_n : \kappa_n$, where each x_i is a variable and κ_i a simple kind, such that $x_1, \ldots, x_n$ are all distinct.*

Definition 4.2 (Erasure) *For a kind A, the* erasure *of A, A^-, is the simple kind defined recursively as follows:*

$$\begin{array}{rcl}(1)^- & \equiv & 1^- \\ (Type)^- & \equiv & Type^- \\ El(t)^- & \equiv & El^- \\ (\Pi x : A.B)^- & \equiv & A^- \to B^- \\ (\Sigma x : A.B)^- & \equiv & A^- \times B^-\end{array}$$

We define the erasure Γ^- of a consistent context Γ by:

$$(x_1 : A_1, \ldots, x_n : A_n)^- \equiv x_1 : A_1^-, \ldots, x_n : A_n^-$$

Lemma 4.3 (Erasure Preservation for Definitional Equality) *If $\Gamma \vdash A = B$ then $A^- \equiv B^-$.*

Lemma 4.4 (Erasure Preservation under Substitution) $([a/x]A)^- \equiv A^-$

We now give an equality relation in the simple type theory, which we shall show is equivalent to definitional equality. We define by mutual recursion the relations $\Delta \vdash a \Leftrightarrow b : \kappa$ and $\Delta \vdash a \leftrightarrow b : \kappa$ among simple contexts Δ, terms a and b, and simple kinds κ. We then define the relation $\Delta \vdash A \Leftrightarrow B$ among simple contexts Δ and kinds A and B. The definition is given in Figure 3.

We shall refer to expressions of the form $a \Leftrightarrow b : A$, $a \leftrightarrow b : A$ and $A \Leftrightarrow B$ as *algorithmic equality judgements*.

We give in Figures 4 and 5 three algorithms that we claim are decision procedures for these relations. We shall shortly prove that, when these algorithms terminate, they correctly decide the relations. We shall not prove that they always terminate when given well-typed terms as input until Section 7.

Lemma 4.5 *If $\Delta \vdash a \leftrightarrow b : \kappa$, then a and b are whnfs.*

Lemma 4.6 (Determinacy of Structural Equality) *If $\Delta \vdash a \leftrightarrow b : \kappa$ and $\Delta \vdash a \leftrightarrow c : \kappa'$, then $\kappa \equiv \kappa'$.*

Lemma 4.7 *Algorithms A, B and C are correct. That is:*

1. *If Algorithm A, given input Δ, a, b and κ, terminates answering yes, then $\Delta \vdash a \Leftrightarrow b : \kappa$. If it terminates answering no, then $\Delta \nvdash a \Leftrightarrow b : \kappa$.*

2. *If Algorithm B, given input Δ, a and b, terminates returning κ, then $\Delta \vdash a \leftrightarrow b : \kappa$. If it terminates answering no, then there does not exist κ such that $\Delta \vdash a \leftrightarrow b : \kappa$.*

3. *If Algorithm C, given input Δ, A and B, terminates answering yes, then $\Delta \vdash A \Leftrightarrow B$. If it terminates answering no, then $\Delta \nvdash A \Leftrightarrow B$.*

Type-directed Term Equality

$$(whrl)\,\frac{\Delta \vdash a' \Leftrightarrow b : \alpha}{\Delta \vdash a \Leftrightarrow b : \alpha}\,(a \rhd_{whd} a') \qquad (whrr)\,\frac{\Delta \vdash a \Leftrightarrow b' : \alpha}{\Delta \vdash a \Leftrightarrow b : \alpha}\,(b \rhd_{whd} b')$$

$$(lift)\,\frac{\Delta \vdash a \leftrightarrow b : \alpha}{\Delta \vdash a \Leftrightarrow b : \alpha} \qquad (\eta_\Pi)\,\frac{\Delta, x : \kappa_1 \vdash f(x) \Leftrightarrow g(x) : \kappa_2}{\Delta \vdash f \Leftrightarrow g : \kappa_1 \to \kappa_2}$$

$$(\eta_\Sigma)\,\frac{\Delta \vdash \pi_1(a) \Leftrightarrow \pi_1(b) : \kappa_1 \quad \Delta \vdash \pi_2(a) \Leftrightarrow \pi_2(b) : \kappa_2}{\Delta \vdash a \Leftrightarrow b : \kappa_1 \times \kappa_2} \qquad (\eta_1)\,\frac{}{\Delta \vdash a \Leftrightarrow b : 1^-}$$

(In the first three clauses, α is either $Type^-$ or El^-.)

Structural Term Equality

$$(Var)\,\frac{}{\Delta \vdash x \leftrightarrow x : \kappa}\,(x : \kappa \in \Delta) \qquad (*)\,\frac{}{\Delta \vdash * \leftrightarrow * : 1^-}$$

$$(app)\,\frac{\Delta \vdash f \leftrightarrow g : \kappa_1 \to \kappa_2 \quad \Delta \vdash a \Leftrightarrow b : \kappa_1}{\Delta \vdash f(a) \leftrightarrow g(b) : \kappa_2}$$

$$(\pi_1)\,\frac{\Delta \vdash a \leftrightarrow b : \kappa_1 \times \kappa_2}{\Delta \vdash \pi_1(a) \leftrightarrow \pi_1(b) : \kappa_1} \qquad (\pi_2)\,\frac{\Delta \vdash a \leftrightarrow b : \kappa_1 \times \kappa_2}{\Delta \vdash \pi_2(a) \leftrightarrow \pi_2(b) : \kappa_2}$$

Algorithmic Kind Equality

$$(Type)\,\frac{}{\Delta \vdash Type \Leftrightarrow Type} \qquad (\Pi)\,\frac{\Delta \vdash A \Leftrightarrow B \quad \Delta, x : A^- \vdash C \Leftrightarrow D}{\Delta \vdash \Pi x : A.C \Leftrightarrow \Pi x : B.D}$$

$$(\Sigma)\,\frac{\Delta \vdash A \Leftrightarrow B \quad \Delta, x : A^- \vdash C \Leftrightarrow D}{\Delta \vdash \Sigma x : A.C \Leftrightarrow \Sigma x : B.D} \qquad (1)\,\frac{}{\Delta \vdash 1 \Leftrightarrow 1}$$

$$(El)\,\frac{\Delta \vdash t \Leftrightarrow u : Type^-}{\Delta \vdash El(t) \Leftrightarrow El(u)}$$

Figure 3: Algorithmic equality

Algorithm A Given a simple context Δ, terms a, b, and a simple kind κ, to decide whether $\Delta \vdash a \Leftrightarrow b : \kappa$.

If $\kappa \equiv Type^-$ or El^-: Weakly head reduce a and b to a' and b', respectively. Call Algorithm B to find τ such that $\Delta \vdash a \leftrightarrow b : \tau$. If $\kappa \equiv \tau$, answer yes; if not, or if no such τ exists, answer no.

If $\kappa \equiv 1^-$: Answer yes.

If $\kappa \equiv \kappa_1 \to \kappa_2$: Let x be a fresh variable. Call Algorithm A to decide whether $\Delta, x : \kappa_1 \vdash f(x) \Leftrightarrow g(x) : \kappa_2$. If so, answer yes; if not, answer no.

If $\kappa \equiv \kappa_1 \times \kappa_2$: Call Algorithm A to decide whether $\Delta \vdash \pi_1(a) \Leftrightarrow \pi_1(b) : \kappa_1$ and $\Delta \vdash \pi_2(a) \Leftrightarrow \pi_2(b) : \kappa_2$. If both of these hold, answer yes; if either fails, answer no.

Algorithm B Given a simple context Δ, and terms a, b, to find a simple kind κ such that $\Delta \vdash a \leftrightarrow b : \kappa$, or answer 'no' if no such κ exists:

If a is a variable: If $b \equiv a$ and $a : \kappa \in \Delta$, return κ. If $b \not\equiv a$, or $a \notin \text{dom } \Delta$, answer no.

If $a \equiv *$: If $b \equiv *$, return 1^-; if $b \not\equiv *$, answer no.

If $a \equiv f(c)$: If b is not of the form $g(d)$, answer no.

If $b \equiv g(d)$, call Algorithm B to find τ such that $\Delta \vdash f \leftrightarrow g : \tau$. If no such τ exists, or if τ is not of the form $\kappa_1 \to \kappa_2$, answer no. If $\tau \equiv \kappa_1 \to \kappa_2$, call Algorithm A to decide whether $\Delta \vdash a \Leftrightarrow b : \kappa_1$. If so, return κ_2; if not, answer no.

If $a \equiv \pi_i(c)$ $(i \equiv 1, 2)$: If b is not of the form $\pi_i(d)$, answer no.

If $b \equiv \pi_i(d)$, call Algorithm B to find τ such that $\Delta \vdash a \leftrightarrow b : \tau$. If no such τ exists, or if τ is not of the form $\kappa_1 \times \kappa_2$, answer no. If $\tau \equiv \kappa_1 \times \kappa_2$, return κ_i.

If $a \equiv \lambda x : A.c$ or $(c, d)_{xB}$: Answer no.

Figure 4: Decision Procedures for Algorithmic Equality

Algorithm C Given a simple context Δ and kinds A, B, to decide whether $\Delta \vdash A \Leftrightarrow B$:

If $A \equiv Type$: If $B \equiv Type$, answer yes; if not, answer no.

If $A \equiv Qx : C.D$ ($Q \equiv \Pi, \Sigma$): Assume (by α-conversion) that $x \notin \mathrm{dom}\ \Delta$.

If B is not of the form $Qx : E.F$, answer no.

If $B \equiv Qx : E.F$, call Algorithm C to decide whether $\Delta \vdash C \Leftrightarrow E$ and $\Delta, x : C^- \vdash D \Leftrightarrow F$. If both of these hold, answer yes; if either fails, answer no.

If $A \equiv 1$: If $B \equiv 1$, answer yes; if not, answer no.

If $A \equiv El(a)$: If B is not of the form $El(b)$, answer no.

If $B \equiv El(b)$, call Algorithm A to decide whether $\Delta \vdash a \Leftrightarrow b : Type^-$. If so, answer yes; if not, answer no.

Figure 5: Decision Procedures for Algorithmic Equality

4.1 Properties of Algorithmic Equality

Lemma 4.8 (Weakening) *For any algorithmic equality judgement J, if $\Delta \vdash J$ and $\Delta \subseteq \Delta'$ then $\Delta' \vdash J$.*

Lemma 4.9 (Contraction) *For any algorithmic equality judgement J, if $\Delta, x_1 : \kappa, x_2 : \kappa, \Delta' \vdash J$ then $\Delta, x : \kappa, \Delta' \vdash [x/x_1][x/x_2]J$.*

Lemma 4.10 (Strengthening) *For any algorithmic equality judgement J, if $\Delta, x : \kappa, \Delta' \vdash J$ and $x \notin FV(J)$ then $\Delta, \Delta' \vdash J$.*

Lemma 4.11 (Symmetry of Algorithmic Equality for Terms) *1. If $\Delta \vdash a \Leftrightarrow b : \kappa$ then $\Delta \vdash b \Leftrightarrow a : \kappa$.*

2. If $\Delta \vdash a \leftrightarrow b : \kappa$ then $\Delta \vdash b \leftrightarrow a : \kappa$.

Lemma 4.12 (Erasure Preservation under Algorithmic Equality) *If $\Delta \vdash A \Leftrightarrow B$ then $A^- \equiv B^-$.*

Lemma 4.13 (Symmetry of Algorithmic Equality for Kinds) *If $\Delta \vdash A \Leftrightarrow B$ then $\Delta \vdash B \Leftrightarrow A$.*

Lemma 4.14 (Transitivity of Algorithmic Equality for Terms) *1. If $\Delta \vdash a \Leftrightarrow b : \kappa$ and $\Delta \vdash b \Leftrightarrow c : \kappa$, then $\Delta \vdash a \Leftrightarrow c : \kappa$.*

2. If $\Delta \vdash a \leftrightarrow b : \kappa$ and $\Delta \vdash b \leftrightarrow c : \kappa$, then $\Delta \vdash a \leftrightarrow c : \kappa$.

Proof The two parts are proved simultaneously, the first by double induction on the two premisses, the second simply by induction on the first premise.

For part 1, the cases that are considered are:

- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from $(whrl)$. (The case $\Delta \vdash b \Leftrightarrow c : \kappa$ came from $(whrr)$ is similar.)
- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from $(lift)$. (The case $\Delta \vdash b \Leftrightarrow c : \kappa$ came from $(lift)$ is similar.)
- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from (η_Π). (The case $\Delta \vdash b \Leftrightarrow c : \kappa$ came from (η_Π) is similar.)
- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from (η_Σ). (The case $\Delta \vdash b \Leftrightarrow c : \kappa$ came from (η_Σ) is similar.)
- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from (η_1). (The case $\Delta \vdash b \Leftrightarrow c : \kappa$ came from (η_1) is similar.)
- $\Delta \vdash a \Leftrightarrow b : \kappa$ came from $(whrr)$ and $\Delta \vdash b \Leftrightarrow c : \kappa$ came from $(whrl)$.

Q.E.D.

Lemma 4.15 (Transitivity of Algorithmic Equality for Kinds) *If* $\Delta \vdash A \Leftrightarrow B$ *and* $\Delta \vdash B \Leftrightarrow C$ *then* $\Delta \vdash A \Leftrightarrow C$.

5 The Kripke Logical Relation

A *substitution* is defined to be a finite sequence of pairs $(a_1/x_1, \ldots, a_n/x_n)$, where each a_i is a term and x_i a variable, such that x_1, ..., x_n are all distinct. We then define the result of applying a substitution σ to a term a, $a\sigma$, or a kind A, $A\sigma$, recursively as follows:

Let $\sigma \equiv (a_1/x_1, \ldots, a_n/x_n)$.

$$
\begin{array}{rcl}
x_i\sigma & \equiv & a_i \quad (i = 1, \ldots, n) \\
z\sigma & \equiv & z \quad (z \notin \{x_1, \ldots, x_n\}) \\
(\lambda y : A.a)\sigma & \equiv & \lambda y : A\sigma.a\sigma \\
(f(a))\sigma & \equiv & f\sigma(a\sigma) \\
(a, b)_{yA}\sigma & \equiv & (a\sigma, b\sigma)_{yA\sigma} \\
\pi_1(a)\sigma & \equiv & \pi_1(a\sigma) \\
\pi_2(a)\sigma & \equiv & \pi_2(a\sigma) \\
*\sigma & \equiv & * \\
(\Pi y : A.B)\sigma & \equiv & \Pi y : A\sigma.B\sigma \\
(\Sigma y : A.B)\sigma & \equiv & \Sigma y : A\sigma.B\sigma \\
1\sigma & \equiv & 1 \\
(Type)\sigma & \equiv & Type \\
El(a)\sigma & \equiv & El(a\sigma)
\end{array}
$$

- $\Delta \vdash a = b \in [\![Type^-]\!]$ iff $\Delta \vdash a \Leftrightarrow b : Type^-$.
- $\Delta \vdash a = b \in [\![El^-]\!]$ iff $\Delta \vdash a \Leftrightarrow b : El^-$.
- $\Delta \vdash a = b \in [\![1^-]\!]$ for any terms a, b.
- $\Delta \vdash f = g \in [\![\kappa_1 \to \kappa_2]\!]$ iff, whenever $\Delta' \supseteq \Delta$ and $\Delta' \vdash a = b \in [\![\kappa_1]\!]$, then $\Delta' \vdash f(a) = g(b) \in [\![\kappa_2]\!]$.
- $\Delta \vdash a = b \in [\![\kappa_1 \times \kappa_2]\!]$ iff $\Delta \vdash \pi_1(a) = \pi_1(b) \in [\![\kappa_1]\!]$ and $\Delta \vdash \pi_2(a) = \pi_2(b) \in [\![\kappa_2]\!]$.
- $\Delta \vdash \sigma = \theta \in [\![\langle\rangle]\!]$ iff $\sigma \equiv \theta \equiv ()$.
- $\Delta \vdash \sigma = \theta \in [\![\Theta, x : \kappa]\!]$ iff $\sigma \equiv (\sigma', a/x)$ and $\theta \equiv (\theta', b/x)$, where $\Delta \vdash \sigma' = \theta' \in [\![\Theta]\!]$ and $\Delta \vdash a = b \in [\![\kappa]\!]$.

Figure 6: The Kripke Logical Relation

where, throughout, we assume $y \notin FV(\sigma)$. (We may make this assumption due to α-convertibility.)

For any consistent context Γ, we define the substitution id_Γ as follows:
If $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$, then $id_\Gamma \equiv (x_1/x_1, \ldots, x_n/x_n)$.

We define a relation $\Delta \vdash a = b \in [\![\kappa]\!]$ among simple contexts Δ, terms a, b and simple types κ by recursion on κ. We then define a relation $\Delta \vdash \sigma = \theta \in [\![\Theta]\!]$ among simple contexts Δ, Θ and substitutions σ, θ, by recursion on Θ. The definitions are given in Figure 6.

We shall refer to expressions of the form $a = b \in [\![\kappa]\!]$ or $\sigma = \theta \in [\![\Delta]\!]$ as *Kripke logical judgements*.

Lemma 5.1 (Weakening) *For any Kripke logical judgement R, if $\Delta \vdash R$ and $\Delta \subseteq \Delta'$, then $\Delta' \vdash R$.*

Lemma 5.2 (Contraction) *For any Kripke logical judgement R, if $\Delta, x_1 : \kappa, x_2 : \kappa, \Delta' \vdash R$, then $\Delta, x : \kappa, \Delta' \vdash [x/x_1][x/x_2]R$.*

Lemma 5.3 (Strengthening) *For any Kripke logical judgement R, if $\Delta, x : \kappa, \Delta' \vdash R$ and $x \notin FV(R)$, then $\Delta, \Delta' \vdash R$.*

Lemma 5.4

1. *If $\Delta \vdash a = b \in [\![\kappa]\!]$, then $\Delta \vdash a \Leftrightarrow b : \kappa$.*
2. *If $\Delta \vdash a \leftrightarrow b : \kappa$, then $\Delta \vdash a = b \in [\![\kappa]\!]$.*

Lemma 5.5 (Head Expansion on Left) *If $\Delta \vdash a' = b \in [\![\kappa]\!]$, and $a \rhd_{whd} a'$, then $\Delta \vdash a = b \in [\![\kappa]\!]$.*

Lemma 5.6 (Head Expansion on Right) *If* $\Delta \vdash a = b' \in [\![\kappa]\!]$, *and* $b \rhd_{whd} b'$, *then* $\Delta \vdash a = b \in [\![\kappa]\!]$.

Lemma 5.7 (Symmetry of Term Relation) *If* $\Delta \vdash a = b \in [\![\kappa]\!]$, *then* $\Delta \vdash b = a \in [\![\kappa]\!]$.

Lemma 5.8 (Symmetry of Substitution Relation) *If* $\Delta \vdash \sigma = \theta \in [\![\Theta]\!]$, *then* $\Delta \vdash \theta = \sigma \in [\![\Theta]\!]$.

Lemma 5.9 (Transitivity of Term Relation) *If* $\Delta \vdash a = b \in [\![\kappa]\!]$ *and* $\Delta \vdash b = c \in [\![\kappa]\!]$, *then* $\Delta \vdash a = c \in [\![\kappa]\!]$.

Lemma 5.10 (Transitivity of Substitution Relation) *If* $\Delta \vdash \sigma = \theta \in [\![\Theta]\!]$ *and* $\Delta \vdash \theta = \delta \in [\![\Theta]\!]$, *then* $\Delta \vdash \sigma = \delta \in [\![\Theta]\!]$.

Lemma 5.11 *If* $\Gamma \vdash a : A$, *and* $\Delta \vdash \sigma = \theta \in [\![\Gamma^-]\!]$, *then* $\Delta \vdash a\sigma = a\theta \in [\![A^-]\!]$

Lemma 5.12 *If* $\Gamma \vdash a = b : A$, *and* $\Delta \vdash \sigma = \theta \in [\![\Gamma^-]\!]$, *then* $\Delta \vdash a\sigma = b\theta \in [\![A^-]\!]$.

Lemma 5.13 *For any consistent context* Γ, $\Gamma^- \vdash id_\Gamma = id_\Gamma \in [\![\Gamma^-]\!]$.

Lemma 5.14 *If* $\Gamma \vdash a = b : A$, *then* $\Gamma^- \vdash a = b \in [\![A^-]\!]$.

Theorem 1 (Soundness Theorem)
1. *If* $\Gamma \vdash a = b : A$ *then* $\Gamma^- \vdash a \Leftrightarrow b : A^-$.
2. *If* $\Gamma \vdash A = B$ *then* $\Gamma^- \vdash A \Leftrightarrow B$.

Proof

1. If $\Gamma \vdash a = b : A$, then

$$\begin{array}{rrll} & \Gamma^- & \vdash & a = b \in [\![A^-]\!] \quad \text{(Lemma 5.14)} \\ \therefore & \Gamma^- & \vdash & a \Leftrightarrow b : A^- \quad \text{(Lemma 5.4)} \end{array}$$

2. We first prove that, if $\Gamma \vdash A$ kind, then $\Gamma \vdash A \Leftrightarrow A$. The proof shall be by induction on $\Gamma \vdash A$ kind.

$(\Pi), (\Sigma)$

$$\frac{\Gamma, x : A \vdash B \text{ kind}}{\Gamma \vdash Qx : A.B \text{ kind}} \, (Q \equiv \Pi, \Sigma)$$

$$\begin{array}{rrll} & \text{There is a subderivation of } \Gamma, x : A \text{ valid} & & \text{(context validity)} \\ \therefore & \text{There is a subderivation of } \Gamma \vdash A \text{ kind} & & \text{(inversion)} \\ \therefore & \Gamma^- & \vdash & A \Leftrightarrow A \quad \text{(i.h.)} \\ & \Gamma^-, x : A^- & \vdash & B \Leftrightarrow B \quad \text{(i.h.)} \\ \therefore & \Gamma^- & \vdash & Qx : A.B \Leftrightarrow Qx : A.B \quad (Q) \end{array}$$

(1)

$$\Gamma^- \vdash 1 \Leftrightarrow 1 \quad (1)$$

$(Type)$

$$\Gamma^- \vdash Type \Leftrightarrow Type \quad (Type)$$

(El)

$$\frac{\Gamma \vdash t : Type}{\Gamma \vdash El(t)\ \text{kind}}$$

$$\begin{array}{rrll} & \Gamma & \vdash & t = t : Type \quad (Refl) \\ \therefore & \Gamma^- & \vdash & t \Leftrightarrow t : Type^- \quad \text{(part 1)} \\ \therefore & \Gamma^- & \vdash & El(t) \Leftrightarrow El(t) \quad (El) \end{array}$$

The main result now follows by induction on $\Gamma \vdash A = B$.

$(KRefl)$ Immediate from the claim above.

$(KSym)$ Follows from symmetry of the Kripke relation.

$(KTrans)$ Follows from transitivity of the Kripke relation.

$(\Pi - eq)$ Follows from the rule (Π).

$(\Sigma - eq)$ Follows from the rule (Σ).

$(El - eq)$ Follows from part 1 and the rule (El).

Q.E.D.

6 The Completeness Theorem

Lemma 6.1 (Subject Reduction) *If $a \rhd_{whd} b$ and $\Gamma \vdash a : A$, then $\Gamma \vdash a = b : A$.*

Proof By induction on $a \rhd_{whd} b$.

Q.E.D.

Theorem 2 (Completeness Theorem)

1. *If $\Gamma \vdash a : A$, $\Gamma \vdash b : A$, and $\Gamma^- \vdash a \Leftrightarrow b : A^-$, then $\Gamma \vdash a = b : A$.*
2. *If $\Gamma \vdash a : A$, $\Gamma \vdash b : B$ and $\Gamma^- \vdash a \leftrightarrow b : \kappa$, then $\Gamma \vdash a = b : A$, $\Gamma \vdash A = B$ and $A^- \equiv B^- \equiv \kappa$.*
3. *If $\Gamma \vdash A$ kind, $\Gamma \vdash B$ kind and $\Gamma^- \vdash A \Leftrightarrow B$, then $\Gamma \vdash A = B$.*

Proof Parts 1 and 2 are proven simultaneously by induction on $\Gamma^- \vdash a \Leftrightarrow b : A^-$ or $\Gamma^- \vdash a \leftrightarrow b : \kappa$.

1. $(whrr)$

$$\frac{a \rhd_{whr} a' \quad \Gamma^- \vdash a' \Leftrightarrow b : A^-}{\Gamma^- \vdash a \Leftrightarrow b : A^-}$$

$$\begin{array}{lllll} & \Gamma & \vdash & a = a' : A & \text{(subject reduction)} \\ \therefore & \Gamma & \vdash & a' : A & \text{(validity)} \\ \therefore & \Gamma & \vdash & a' = b : A & \text{(i.h.)} \\ \therefore & \Gamma & \vdash & a = b : A & (Trans) \end{array}$$

$(whrl)$ Similar.

$(lift)$ Immediate from i.h.

(η_Π)

$$\frac{\Gamma^-, x : B^- \vdash a(x) \Leftrightarrow b(x) : C^-}{\Gamma^- \vdash a \Leftrightarrow b : B^- \to C^- \quad (A \equiv \Pi x : B.C)}$$

$$\begin{array}{llllll} & & \Gamma & \vdash & a : \Pi x : B.C & \\ \therefore & & \Gamma & \vdash & \Pi x : B.C \text{ kind} & \text{(validity)} \\ \therefore & \Gamma, x : B & \vdash & & C \text{ kind} & \text{(inversion)} \\ \therefore & \Gamma, x : B & \text{valid} & & & \text{(context validity)} \\ \therefore & \Gamma, x : B & \vdash & & a : \Pi x : B.C & \text{(weakening)} \\ & \Gamma, x : B & \vdash & & x : B & (Var) \\ \therefore & \Gamma, x : B & \vdash & & a(x) : C & (app) \\ & \Gamma, x : B & \vdash & & b : \Pi x : B.C & \text{(weakening)} \\ \therefore & \Gamma, x : B & \vdash & & b(x) : C & (app) \\ \therefore & \Gamma, x : B & \vdash & & a(x) = b(x) : C & \text{(i.h.)} \\ & & \Gamma & \vdash & B \text{ kind} & \text{(inverting } \Gamma, x : A \text{ valid)} \\ \therefore & & \Gamma & \vdash & B = B & (KRefl) \\ \therefore & & \Gamma & \vdash & \lambda x : B.a(x) = \lambda x : B.b(x) : \Pi x : B.C & (\lambda - eq) \\ & & \Gamma & \vdash & \lambda x : B.a(x) = a : \Pi x : B.C & (\eta_\Pi) \\ \therefore & & \Gamma & \vdash & a = \lambda x : B.a(x) : \Pi x : B.C & (Sym) \\ \therefore & & \Gamma & \vdash & a = \lambda x : B.b(x) : \Pi x : B.C & (Trans) \\ & & \Gamma & \vdash & \lambda x : B.b(x) = b : \Pi x : B.C & (\eta_\Pi) \\ \therefore & & \Gamma & \vdash & a = b : \Pi x : B.C & (Trans) \end{array}$$

(η_Σ)

$$\frac{\Gamma^- \vdash \pi_1(a) \Leftrightarrow \pi_1(b) : B^- \quad \Gamma^- \vdash \pi_2(a) \Leftrightarrow \pi_2(b) : C^-}{\Gamma^- \vdash a \Leftrightarrow b : B^- \times C^- \quad (A \equiv \Sigma x : B.C)}$$

$$
\begin{array}{lrcll}
 & \Gamma & \vdash & a : \Sigma x : B.C & \\
\therefore & \Gamma & \vdash & \pi_1(a) : B & (\pi_1) \\
 & \Gamma & \vdash & b : \Sigma x : B.C & \\
\therefore & \Gamma & \vdash & \pi_1(b) : B & (\pi_1) \\
\therefore & \Gamma & \vdash & \pi_1(a) = \pi_1(b) : B & (\text{i.h.}) \\
 & \Gamma & \vdash & \pi_2(a) : [\pi_1(a)/x]C & (\pi_2) \\
 & \Gamma & \vdash & \pi_2(b) : [\pi_1(b)/x]C & (\pi_2) \\
 & \Gamma & \vdash & \Sigma x : B.C \text{ kind} & (\text{validity}) \\
\therefore & \Gamma, x : B & \vdash & C \text{ kind} & (\text{inversion}) \\
\therefore & \Gamma & \vdash & [\pi_1(a)/x]C = [\pi_1(b)/x]C & (\text{functionality}) \\
\therefore & \Gamma & \vdash & [\pi_1(b)/x]C = [\pi_1(a)/x]C & (KSym) \\
\therefore & \Gamma & \vdash & \pi_2(b) : [\pi_1(a)/x]C & (Eq) \\
\therefore & \Gamma & \vdash & \pi_2(a) = \pi_2(b) : [\pi_1(a)/x]C & (\text{i.h.}) \\
 & \Gamma, x : B & \vdash & C = C & (KRefl) \\
\therefore & \Gamma & \vdash & (\pi_1(a), \pi_2(a))_{xC} = (\pi_1(b), \pi_2(b))_{xC} : \Sigma x : B.C & (pair - eq) \\
 & \Gamma & \vdash & (\pi_1(a), \pi_2(a))_{xC} = a : \Sigma x : B.C & (\eta_\Sigma) \\
\therefore & \Gamma & \vdash & a = (\pi_1(a), \pi_2(a))_{xC} : \Sigma x : B.C & (Sym) \\
\therefore & \Gamma & \vdash & a = (\pi_1(b), \pi_2(b))_{xC} : \Sigma x : B.C & (Trans) \\
 & \Gamma & \vdash & (\pi_1(b), \pi_2(b))_{xC} = b : \Sigma x : B.C & (\eta_\Sigma) \\
\therefore & \Gamma & \vdash & a = b : \Sigma x : B.C & (Trans)
\end{array}
$$

(η_1)

$$
\frac{}{\Gamma^- \vdash a \Leftrightarrow b : 1^-}
$$

$$
\begin{array}{lrcll}
 & \Gamma & \vdash & a : 1 & \\
\therefore & \Gamma & \vdash & * = a : 1 & (\eta_1) \\
\therefore & \Gamma & \vdash & a = * : 1 & (Sym) \\
 & \Gamma & \vdash & b : 1 & \\
\therefore & \Gamma & \vdash & * = b : 1 & (\eta_1) \\
\therefore & \Gamma & \vdash & a = b : 1 & (Trans)
\end{array}
$$

2. (Var)

$$
\frac{}{\Gamma^- \vdash x \leftrightarrow x : C^-} \, (x : C \in \Gamma)
$$

$$
\begin{array}{lrcll}
 & \Gamma & \vdash & x = x : A & (Refl) \\
 & \Gamma & \vdash & A = C & (\text{typing inversion}) \\
 & \Gamma & \vdash & B = C & (\text{typing inversion}) \\
\therefore & \Gamma & \vdash & C = B & (KSym) \\
\therefore & \Gamma & \vdash & A = B & (KTrans) \\
 & A^- \equiv B^- \equiv C^- & & & (\text{erasure preservation})
\end{array}
$$

$(*)$

$$
\frac{}{\Gamma^- \vdash * \leftrightarrow * : 1^-}
$$

$$
\begin{array}{rrcll}
 & \Gamma & \vdash & * = * : A & (Refl) \\
 & \Gamma & \vdash & 1 = A & \text{(typing inversion)} \\
\therefore & \Gamma & \vdash & A = 1 & (KSym) \\
 & \Gamma & \vdash & 1 = B & \text{(typing inversion)} \\
\therefore & \Gamma & \vdash & A = B & (KTrans) \\
 & A^- \equiv B^- \equiv 1^- & & & \text{(erasure preservation)}
\end{array}
$$

(app)

$$\frac{\Gamma^- \vdash f \leftrightarrow g : \kappa_1 \to \kappa_2 \quad \Gamma^- \vdash a \Leftrightarrow b : \kappa_1}{\Gamma^- \vdash f(a) \leftrightarrow g(b) : \kappa_2}$$

By typing inversion, there exist A_1, A_2, B_1, B_2 such that

$$
\begin{array}{rrcll}
 & \Gamma & \vdash & f : \Pi x : A_1 . A_2 & \\
 & \Gamma & \vdash & a : A_1 & \\
 & \Gamma & \vdash & [a/x]A_2 = A & \\
 & \Gamma & \vdash & g : \Pi x : B_1 . B_2 & \\
 & \Gamma & \vdash & b : B_1 & \\
 & \Gamma & \vdash & [b/x]B_2 = B & \\
\therefore & \Gamma & \vdash & f = g : \Pi x : A_1 . A_2 & \text{(i.h.)} \\
 & \Gamma & \vdash & \Pi x : A_1 . A_2 = \Pi x : B_1 . B_2 & \text{(i.h.)} \\
 & A_1^- \to A_2^- \equiv B_1^- \to B_2^- \equiv \kappa_1 \to \kappa_2 & & & \text{(i.h.)} \\
\therefore & A^- \equiv ([a/x]A_2)^- \equiv A_2^- \equiv \kappa_2 & & & \text{(erasure preservation)} \\
 & B^- \equiv ([b/x]B_2)^- \equiv B_2^- \equiv \kappa_2 & & & \text{(erasure preservation)} \\
 & \Gamma & \vdash & A_1 = B_1 & \text{(equality inversion)} \\
\therefore & \Gamma & \vdash & B_1 = A_1 & (Sym) \\
\therefore & \Gamma & \vdash & b : A_1 & (Eq) \\
 & A_1^- & \equiv & \kappa_1 & \\
\therefore & \Gamma & \vdash & a = b : A_1 & \text{(i.h.)} \\
\therefore & \Gamma & \vdash & f(a) = g(b) : [a/x]A_2 & (app - eq) \\
\therefore & \Gamma & \vdash & f(a) = g(b) : A & (= R) \\
 & \Gamma, x : A_1 & \vdash & A_2 = B_2 & \text{(equality inversion)} \\
\therefore & \Gamma & \vdash & [a/x]A_2 = [b/x]B_2 & \text{(functionality)} \\
 & \Gamma & \vdash & A = [a/x]A_2 & (KSym) \\
\therefore & \Gamma & \vdash & A = [b/x]B_2 & (KTrans) \\
\therefore & \Gamma & \vdash & A = B & (KTrans)
\end{array}
$$

(π_1)

$$\frac{\Gamma^- \vdash a \leftrightarrow b : \kappa_1 \times \kappa_2}{\Gamma^- \vdash \pi_1(a) \leftrightarrow \pi_1(b) : \kappa_1}$$

By typing inversion, there exist C, D such that

$$\begin{array}{lrll}
 & \Gamma & \vdash & a : \Sigma x : A.C & \\
 & \Gamma & \vdash & b : \Sigma x : B.D & \\
\therefore & \Gamma & \vdash & a = b : \Sigma x : A.C & \text{(i.h.)} \\
 & \Gamma & \vdash & \Sigma x : A.C = \Sigma x : B.D & \text{(i.h.)} \\
 & A^- \times C^- \equiv B^- \times D^- \equiv \kappa_1 \times \kappa_2 & & & \text{(i.h.)} \\
\therefore & \Gamma & \vdash & \pi_1(a) = \pi_1(b) : A & (\pi_1 - eq) \\
 & \Gamma & \vdash & A = B & \text{(equality inversion)} \\
 & & & A^- \equiv B^- \equiv \kappa_1 &
\end{array}$$

(π_2)

$$\frac{\Gamma^- \vdash a \leftrightarrow b : \kappa_1 \times \kappa_2}{\Gamma^- \vdash \pi_2(a) \leftrightarrow \pi_2(b) : \kappa_2}$$

By typing inversion, there exist C, D, E, F such that

$$\begin{array}{lrll}
 & \Gamma & \vdash & a : \Sigma x : C.D & \\
 & \Gamma & \vdash & [\pi_1(a)/x]D = A & \\
 & \Gamma & \vdash & b : \Sigma x : E.F & \\
 & \Gamma & \vdash & [\pi_1(b)/x]F = B & \\
\therefore & \Gamma & \vdash & a = b : \Sigma x : C.D & \text{(i.h.)} \\
 & \Gamma & \vdash & \Sigma x : C.D = \Sigma x : E.F & \text{(i.h.)} \\
 & C^- \times D^- \equiv E^- \times F^- \equiv \kappa_1 \times \kappa_2 & & & \text{(i.h.)} \\
\therefore & \Gamma & \vdash & \pi_2(a) = \pi_2(b) : [\pi_1(a)/x]D & (\pi_2 - eq) \\
\therefore & \Gamma & \vdash & \pi_2(a) = \pi_2(b) : A & (= R) \\
 & \Gamma & \vdash & \pi_1(a) = \pi_1(b) : C & (\pi_1 - eq) \\
 & \Gamma, x : C & \vdash & D = F & \text{(equality inversion)} \\
\therefore & \Gamma & \vdash & [\pi_1(a)/x]D = [\pi_1(b)/x]F & \text{(functionality)} \\
 & \Gamma & \vdash & A = [\pi_1(a)/x]D & (KSym) \\
\therefore & \Gamma & \vdash & A = [\pi_1(b)/x]F & (KTrans) \\
\therefore & \Gamma & \vdash & A = B & (KTrans) \\
 & A^- \equiv ([\pi_1(a)/x]D)^- \equiv D^- \equiv \kappa_2 & & & \text{(erasure preservation)} \\
 & & & B^- \equiv A^- \equiv \kappa_2 & \text{(erasure preservation)}
\end{array}$$

3. The proof shall be by induction on $\Gamma^- \vdash A \Leftrightarrow B$.

$(Type)$

$$\frac{}{\Gamma^- \vdash Type \Leftrightarrow Type}$$

$$\Gamma \quad \vdash \quad Type = Type \quad (KRefl)$$

$(\Pi), (\Sigma)$

$$\frac{\Gamma^- \vdash A \Leftrightarrow B \quad \Gamma^-, x : A^- \vdash C \Leftrightarrow D}{\Gamma^- \vdash Qx : A.C \Leftrightarrow Qx : B.D}\,(Q \equiv \Pi, \Sigma)$$

$$
\begin{array}{lllll}
 & \Gamma, x : A & \vdash & C \text{ kind} & \text{(inversion)} \\
\therefore & \Gamma, x : A & & \text{valid} & \text{(context validity)} \\
\therefore & \Gamma & \vdash & A \text{ kind} & \text{(inversion)} \\
 & \Gamma, x : B & \vdash & D \text{ kind} & \text{(inversion)} \\
 & \Gamma & \vdash & B \text{ kind} & \text{(similarly)} \\
\therefore & \Gamma & \vdash & A = B & \text{(i.h.)} \\
\therefore & \Gamma, x : A & \vdash & D \text{ kind} & \text{(context conversion)} \\
\therefore & \Gamma, x : A & \vdash & C = D & \text{(i.h.)} \\
\therefore & \Gamma & \vdash & Qx : A.C = Qx : B.D & (Q - eq)
\end{array}
$$

(1)

$$
\frac{}{\Gamma^- \vdash 1 \Leftrightarrow 1}
$$

$$
\Gamma \quad \vdash \quad 1 = 1 \quad \text{(KRefl)}
$$

(El)

$$
\frac{\Gamma^- \vdash t \Leftrightarrow u : Type^-}{\Gamma^- \vdash El(t) \Leftrightarrow El(u)}
$$

$$
\begin{array}{lllll}
 & \Gamma & \vdash & t : Type & \text{(inversion)} \\
 & \Gamma & \vdash & u : Type & \text{(inversion)} \\
\therefore & \Gamma & \vdash & t = u : Type & \text{(part 1)} \\
\therefore & \Gamma & \vdash & El(t) = El(u) & (El - eq)
\end{array}
$$

Q.E.D.

Corollary 3 *If* $\Gamma \vdash a : A$, $\Gamma \vdash b : A$, *and* $\Gamma^- \vdash a = b \in [\![A^-]\!]$, *then* $\Gamma \vdash a = b : A$.

7 Decidability

Lemma 7.1 (Decidability for Normalising Terms) *1. If* $\Delta \vdash a \Leftrightarrow a' : \kappa$ *and* $\Delta \vdash b \Leftrightarrow b' : \kappa$, *then Algorithm A terminates given input* Δ, a, b *and* κ.

2. If $\Delta \vdash a \leftrightarrow a' : \kappa_1$ *and* $\Delta \vdash b \leftrightarrow b' : \kappa_2$, *then Algorithm B terminates given input* Δ, a *and* b.

Proof The proof is simultaneous, by double induction on $\Delta \vdash a \Leftrightarrow a' : \kappa$ and $\Delta \vdash b \Leftrightarrow b' : \kappa$ in part 1, and simple induction on $\Delta \vdash a \leftrightarrow a' : \kappa_1$ in part 2.

In part 1, the cases to be considered are:

- $\Delta \vdash a \Leftrightarrow a' : \kappa$ came from $(whrr)$. (The case $\Delta \vdash b \Leftrightarrow b' : \kappa$ came from $(whrr)$ is similar.)
- $\Delta \vdash a \Leftrightarrow a' : \kappa$ came from $(whrl)$. (The case $\Delta \vdash b \Leftrightarrow b' : \kappa$ came from $(whrl)$ is similar.)

- Both premises came from $(lift)$.
- Both premises came from (η_Σ).
- Both premises came from (η_Π).
- Both premises came from (η_1).

Q.E.D.

Lemma 7.2 (Decidability for Normalising Kinds) *If* $\Delta \vdash A \Leftrightarrow A'$ *and* $\Delta \vdash B \Leftrightarrow B'$*, then Algorithm C terminates given input* Δ*,* A *and* B*.*

Lemma 7.3 (Decidability of Algorithmic Equality for Terms) *If* $\Gamma \vdash a : A$ *and* $\Gamma \vdash b : A$*, then Algorithm A terminates given input* Γ^-*,* a*,* b *and* A^-*.*

Lemma 7.4 (Decidability of Algorithmic Equality for Kinds) *If* $\Gamma \vdash A$ kind *and* $\Gamma \vdash B$ kind*, then Algorithm C terminates given input* Γ^-*,* A *and* B*.*

Lemma 7.5 (Decidability of Equality for Terms) *If* $\Gamma \vdash a : A$ *and* $\Gamma \vdash b : A$*, it is decidable whether* $\Gamma \vdash a = b : A$*.*

Lemma 7.6 (Decidability of Equality for Kinds) *If* $\Gamma \vdash A$ kind *and* $\Gamma \vdash B$ kind*, it is decidable whether* $\Gamma \vdash A = B$*.*

We now abandon $LF + \Sigma$, and work only in $LF + \Sigma_s$. We define a relation $\Gamma \vdash a \in A$ among contexts Γ, terms a and kinds A, as well as a relation $\Gamma \vdash A \in$ kind among contexts Γ and kinds A, simultaneously by recursion. The definition is given in Figure 7.

Lemma 7.7 (Completeness of Type Checking) *1. If* Γ valid *and* $\Gamma \vdash a \in A$ *then* $\Gamma \vdash a : A$*.*

2. If Γ valid *and* $\Gamma \vdash A \in$ kind *then* $\Gamma \vdash A$ kind*.*

Lemma 7.8 (Soundness of Type Checking) *1. If* $\Gamma \vdash a : A$*, then there exists* A' *such that* $\Gamma \vdash a \in A'$ *and* $\Gamma \vdash A = A'$*.*

2. If $\Gamma \vdash A$ kind *then* $\Gamma \vdash A \in$ kind*.*

Theorem 4 (Decidability Theorem) $LF + \Sigma_s$ *is decidable.*

Proof The following algorithm, given a judgement J, decides whether J is derivable in $LF + \Sigma$ or not.

To decide whether Γ valid:

If $\Gamma \equiv \langle\rangle$, then Γ valid.

If $\Gamma \equiv \Delta, x : A$, decide whether $\Delta \vdash A$ kind. If so, Γ valid; if not, Γ is not valid.

Terms

$$(Var)\ \frac{}{\Gamma \vdash x \in A}\ (x : A \in \Gamma)$$

$$(app)\ \frac{\Gamma \vdash f \in \Pi x : A_2' : A_1 \quad \Gamma \vdash a \in A_2 \quad \Gamma^- \vdash A_2' \Leftrightarrow A_2}{\Gamma \vdash f(a) \in [a/x]A_1}$$

$$(\lambda)\ \frac{\Gamma \vdash A_1 \in \text{ kind} \quad \Gamma, x : A_1 \vdash a \in A_2}{\Gamma \vdash \lambda x : A_1.a \in \Pi x : A_1.A_2}$$

$$(pair)\ \frac{\Gamma \vdash a \in A \quad \Gamma \vdash b \in B' \quad \Gamma^- \vdash [a/x]B \Leftrightarrow B' \quad \Gamma, x : A \vdash B \in \text{ kind}}{\Gamma \vdash (a, b)_{xB} \in \Sigma x : A.B}$$

$$(\pi_1)\ \frac{\Gamma \vdash a \in \Sigma x : A.B}{\Gamma \vdash \pi_1(a) \in A} \qquad (\pi_2)\ \frac{\Gamma \vdash a \in \Sigma x : A.B}{\Gamma \vdash \pi_2(a) \in [\pi_1(a)/x]B}$$

$$(*)\ \frac{}{\Gamma \vdash * \in 1}$$

Kinds

$$(\Pi)\ \frac{\Gamma \vdash A \in \text{ kind} \quad \Gamma, x : A \vdash B \in \text{ kind}}{\Gamma \vdash \Pi x : A.B \in \text{ kind}}$$

$$(\Sigma)\ \frac{\Gamma \vdash A \in \text{ kind} \quad \Gamma, x : A \vdash B \in \text{ kind}}{\Gamma \vdash \Sigma x : A.B \in \text{ kind}}$$

$$(1)\ \frac{}{\Gamma \vdash 1 \in \text{ kind}} \qquad (Type)\ \frac{}{\Gamma \vdash Type \in \text{ kind}}$$

$$(El)\ \frac{\Gamma \vdash t \in A \quad \Gamma^- \vdash A \Leftrightarrow Type}{\Gamma \vdash El(t) \in \text{ kind}}$$

Figure 7: Algorithmic Type Checking

To decide whether $\Gamma \vdash A$ kind:

Decide whether Γ valid. If not, then $\Gamma \nvdash A$ kind.

If Γ valid, decide whether $\Gamma \vdash A \in$ kind. If so, then $\Gamma \vdash A$ kind; if not, $\Gamma \nvdash A$ kind.

To decide whether $\Gamma \vdash A = B$:

Decide whether $\Gamma \vdash A$ kind and $\Gamma \vdash B$ kind. If either of these fails to hold, then $\Gamma \nvdash A = B$.

If both of these hold, decide whether $\Gamma^- \vdash A \Leftrightarrow B$. If so, then $\Gamma \vdash A = B$; if not, $\Gamma \nvdash A = B$.

To decide whether $\Gamma \vdash a : A$:

Decide whether Γ valid. If not, then $\Gamma \nvdash a : A$.

If Γ valid, find A' such that $\Gamma \vdash a \in A'$. If no such A' exists, then $\Gamma \nvdash a : A$.

Given such an A', decide whether $\Gamma \vdash A = A'$. If so, $\Gamma \vdash a : A$; if not, $\Gamma \nvdash a : A$.

To decide whether $\Gamma \vdash a = b : A$:

Decide whether $\Gamma \vdash a : A$ and $\Gamma \vdash b : A$. If either of these does not hold, then $\Gamma \nvdash a = b : A$.

If both of these hold, decide whether $\Gamma^- \vdash a \Leftrightarrow b : A^-$. If so, then $\Gamma \vdash a = b : A$; if not, $\Gamma \nvdash a = b : A$.

When the first two procedures call one another, the length of the judgement being tested decreases. The third procedure only calls the second; the fourth, only the first and third; and the fifth, only the fourth. Therefore, the algorithm always terminates.

Q.E.D.

8 A Bit of Dessert

Lemma 8.1 (Strengthening for Type Checking) *If* $\Gamma, x : A, \Gamma' \vdash a \in B$ *and* $x \notin FV(\Gamma') \cup FV(a) \cup FV(B)$*, then* $\Gamma, \Gamma' \vdash a \in B$*.*

Corollary 5 (Strengthening) *If* $\Gamma, x : A, \Gamma' \vdash J$ *and* $x \notin FV(\Gamma') \cup FV(J)$*, then* $\Gamma, \Gamma' \vdash J$*.*

References

[1] Thierry Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

[2] Healfdene Goguen. Soundness of typed operational semantics for the logical framework. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99*, volume 1581 of *Lecture Notes in Computer Science*, pages 177–197. Springer, April 1999.

[3] Robert Harper and Frank Pfening. On equivalence and canonical forms in the LF type theory. Technical Report CMU-CS-00-148, Carnegie Mellon University, Pittsburgh, PA, July 2000.

[4] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Number 11 in International Series of Monographs on Computer Science. Oxford University Press, 1994.