Justifying Algorithms for $\beta\eta$ -Conversion

Healfdene Goguen

AT&T Labs, 180 Park Ave., Florham Park NJ 07932 USA

hhg@att.com.

Abstract. Deciding the typing judgement of type theories with dependent types such as the Logical Framework relies on deciding the equality judgement for the same theory. Implementing the conversion algorithm for $\beta\eta$ -equality and justifying this algorithm is therefore an important problem for applications such as proof assistants and modules systems. This article gives a proof of decidability, correctness and completeness of the conversion algorithms for $\beta\eta$ -equality defined by Coquand [3] and Harper and Pfenning [8] for the Logical Framework, relying on established metatheoretic results for the type theory. Proofs are also given of the same properties for a typed algorithm for conversion for System F, a new result.

1 Introduction

In this article we study the decidability of algorithms for $\beta\eta$ -conversion for type theories. We consider two algorithms for the Logical Framework not immediately modeled by reduction to a common $\beta\eta$ -normal form: Coquand's untyped algorithm relating syntactically distinct β -weak-head normal forms [3], and Harper and Pfenning's type-based algorithm [8]. We demonstrate that these algorithms can be shown correct, complete and decidable based on standard metatheoretic properties of type theory, such as strong normalization of $\beta\eta$ -reduction, subject reduction, injectivity of the type constructor Π , and so on. We then apply the same technique to the polymorphic λ -calculus System F.

The focus of many existing developments of the metatheory of type theories with $\beta\eta$ -equality has been on decidability of typechecking, without concern for algorithms for conversion. The fact that the standard metatheory is sufficient to justify the decidability of algorithms has never been demonstrated as far as we are aware, even for Coquand's simple syntactic algorithm. With Harper and Pfenning's definition of a more complex algorithm for equality based on type information, it has become more important to show that the traditional approach to the metatheory of type theories, justifying termination and Church-Rosser for the reduction relation, can be used to show decidability for algorithms that are more complex than the simple comparison of normal forms.

We believe that it may be more efficient and uniform to justify algorithms for $\beta\eta$ -conversion through the traditional approach to metatheory than by studying the algorithm directly. As an example, our proof of the termination of the algorithm for the Logical Framework only requires a single logical relation, as

opposed to the two logical relations used in [11]. Similarly, Harper and Pfenning's approach has not been extended to systems with polymorphism, whereas we are able to adapt our proof straightforwardly to System F. Any approach to the metatheory of type theory with $\beta\eta$ -equality is sufficient, and several methods already exist. Geuvers [4], Goguen [6, 7] and Salvesen [9] all have different approaches to the difficulties presented by η .

The key to all three proofs of decidability is a simple length measure $|-|$ on normal forms of terms, where the value of the measure for an abstraction in normal form, $|\lambda x:A.M|$, is greater than the value of the measure for an application to a variable in normal form, $|M(x)|$. In Coquand's algorithm, an abstraction $\lambda x:A.M$ and a weak-head normal term $y(N_1 \dots N_n)$ are related if M and $y(N_1 \dots N_n, x)$ are related; by our measure, the combined length of the conclusion, $|\lambda x:A.M| + |y(N_1 \dots N_n)|$, is greater than the combined length of the premisses, $|M| + |y(N_1 \dots N_n, x)|$. This same idea can be translated to Harper and Pfenning's type-directed algorithm for conversion.

The remainder of this paper is structured as follows. Section 2 introduces the syntax and standard metatheory for the Logical Framework. Section 3 justifies Coquand's algorithm using the standard metatheory. Section 4 justifies Harper and Pfenning's algorithm using a similar approach. Section 5 presents a type-directed algorithm for conversion for System F and justifies this algorithm. We draw conclusions and discuss future work in Section 6.

2 The Logical Framework

In this section we give our presentation of the Logical Framework. Although our system includes dependent types, we do not refer to this as the Edinburgh Logical Framework or the Martin-Löf Logical Framework, because for simplicity our presentation does not include higher-order kinds and hence does not formally correspond to either system. Otherwise, our system is largely similar to Harper and Pfenning's, but we use a term structure inspired by PTS-style presentations of type theories [2] to take advantage of the similarity of rules in the algorithm.

2.1 Syntax

We assume an infinite collection of variables $x, y, z \in V$. The language of terms and contexts is defined by the following grammar.

$$\begin{aligned} \Gamma \in C &::= () \mid \Gamma, x:A \\ s \in S &::= \text{type} \mid \text{kind} \\ M, N, P, A, B \in T &::= s \mid x \mid \lambda x:A.M \mid M(N) \mid \Pi x:A.B \end{aligned}$$

We say a term is basic if it is a variable x or a sort s , and a term is canonical if it is of the form $\lambda x:A.M$ or $\Pi x:A_1.A_2$. Substitution, $[N/x]M$, is defined as usual for terms, with the obvious extension to contexts. We identify terms and contexts up to α -equivalence, and write $\text{FV}(M)$ for the free variables in M . Let

$\Gamma = x_1:A_1, \dots, x_n:A_n$; then $\text{dom}(\Gamma) \equiv \{x_1, \dots, x_n\}$, and $\Gamma(x)$ is the partial function that returns A_i if $x = x_i$ for some $1 \leq i \leq n$.

2.2 Judgements and Derivations

Our presentation of the Logical Framework has judgements $\Gamma \vdash M : A$ and $\Gamma \vdash M = N : A$. We write $\Gamma \vdash M, N : A$ for $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$, and $\Gamma \vdash J$ to denote either judgement. The rules of inference for typing are given in Figure 1; the rules of inference for the equality judgement are the evident typed compatible closure and least equivalence relation containing the rules β and Ext.

$$\begin{array}{c}
 \text{(type)} \quad () \vdash \text{type} : \text{kind} \quad \text{(Weak)} \quad \frac{\Gamma \vdash A : \text{type}}{\Gamma, x:A \vdash \text{type} : \text{kind}} \quad (x:A \notin \Gamma) \\
 \text{(Var)} \quad \frac{\Gamma \vdash \text{type} : \text{kind}}{\Gamma \vdash x : A} \quad (x:A \in \Gamma) \\
 \text{(}\Pi\text{)} \quad \frac{\Gamma \vdash A_1 : \text{type} \quad \Gamma, x:A_1 \vdash A_2 : s}{\Gamma \vdash \Pi x:A_1.A_2 : s} \quad (s \in \{\text{type}, \text{kind}\}) \\
 \text{(}\lambda\text{)} \quad \frac{\Gamma, x:A_1 \vdash M : A_2}{\Gamma \vdash \lambda x:A_1.M : \Pi x:A_1.A_2} \quad \text{(App)} \quad \frac{\Gamma \vdash M_1 : \Pi x:A_1.A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash M_1(M_2) : [M_2/x]A_2} \\
 \text{(Eq)} \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash A = B : s}{\Gamma \vdash M : B}
 \end{array}$$

Fig. 1. Typing for the Logical Framework

$$\begin{array}{c}
 \text{(}\beta\text{)} \quad \frac{\Gamma \vdash \lambda x:A_1.M : \Pi x:A_1.A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash (\lambda x:A_1.M)(M_2) = [M_2/x]M : [M_2/x]A_2} \\
 \text{(Ext)} \quad \frac{\Gamma, x:A_1 \vdash M(x) = N(x) : A_2 \quad \Gamma \vdash M, N : \Pi x:A_1.A_2}{\Gamma \vdash M = N : \Pi x:A_1.A_2}
 \end{array}$$

2.3 Untyped Reduction

We define reduction $M \rightarrow_{\beta\eta} N$ as the compatible closure of rules β and η :

$$\begin{array}{c}
 (\lambda x:A.M)(N) \beta [N/x]M \\
 \lambda x:A.(M(x)) \eta M \quad (x \notin \text{FV}(M))
 \end{array}$$

Weak-head reduction $\rightarrow_w$ is defined by the following rules:

$$\text{(}\beta\text{)} \quad (\lambda x:A.M)(N) \rightarrow_w [N/x]M \quad \text{(App)} \quad \frac{M \rightarrow_w P}{M(N) \rightarrow_w P(N)}$$

Definition 1 (Normal Forms and Weak-Head Normal Forms). *The β -normal forms are defined inductively as follows: basic terms s and x are normal; abstractions $\lambda x:A.M$ are normal if A and M are normal; products $\Pi x:A_1.A_2$ are normal if A_1 and A_2 are normal; and applications $M_1(M_2)$ are normal if M_1 and M_2 are normal and M_1 is not an abstraction.*

The weak-head normal forms are presented inductively as follows: basic terms s and x are weak-head normal; canonical terms $\lambda x:A.M$ and $\Pi x:A_1.A_2$ are weak-head normal; and applications $M_1(M_2)$ are weak-head normal if M_1 is weak-head normal and not an abstraction.

We write M^{nf} for the β -normal form of M and M^{wnf} for the weak-head normal form of M .

The following definitions apply to reduction relations $\rightarrow_\beta$, $\rightarrow_{\beta\eta}$ and $\rightarrow_w$: we write $\twoheadrightarrow$ for the reflexive, transitive closure of $\rightarrow$, $M \twoheadrightarrow N$ if $M \rightarrow N$ and N is normal, and $M \downarrow N$ if there is a P such that $M \twoheadrightarrow P$ and $N \twoheadrightarrow P$.

Lemma 1.

- If M is normal then there is no N such that $M \rightarrow_\beta N$.
- If M is weak-head normal then there is no N such that $M \rightarrow_w N$.
- Any term M is either weak-head normal or there is an N such that $M \rightarrow_w N$.

2.4 Properties of the Logical Framework

We assume all of the standard properties of the Logical Framework: as we mentioned in the introduction, any approach to proving them is acceptable for the purposes of this article. We state the properties needed here for reference.

Proposition 1 (Generation). *Every derivation of a term is an application of the unique rule of inference for that term followed by a sequence of uses of Eq.*

For example, suppose $\Gamma \vdash \lambda x:A_1.M_0 : A$; then $\Gamma, x:A_1 \vdash M_0 : A_2$ and $\Gamma \vdash \Pi x:A_1.A_2 = A : s$ for some A_2 and s .

Proposition 2.

1. *Free Variables.* If $\Gamma \vdash M : A$ then $\text{FV}(M) \cup \text{FV}(A) \subseteq \text{dom}(\Gamma)$.
2. *Context Validity.* If $\Gamma \vdash J$ then $\Gamma \vdash \text{type} : \text{kind}$.
3. *Thinning.* If $\Gamma, \Gamma' \vdash J$, $x \notin \text{dom}(\Gamma, \Gamma')$ and $\Gamma \vdash A : \text{type}$ then $\Gamma, x:A, \Gamma' \vdash J$.
4. *Substitution.* If $\Gamma, x:A, \Gamma' \vdash J$ and $\Gamma \vdash N : A$ then $\Gamma, [N/x]\Gamma' \vdash [N/x]J$.
5. *Type Correctness.* If $\Gamma \vdash M : A$ then $\Gamma \vdash A : s$ for some s .
6. *Splitting.* If $\Gamma \vdash M = N : A$ then $\Gamma \vdash M, N : A$.
7. *Uniqueness of Types.* If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $\Gamma \vdash A = B : s$ or $A = s$ and $B = s$ for some s .
8. *Context Replacement.* $\Gamma, x:A, \Gamma' \vdash J$ and $\Gamma \vdash A = B : s$ imply $\Gamma, x:B, \Gamma' \vdash J$.
9. *Church–Rosser.* If $\Gamma \vdash M = N : A$ then $M \downarrow_{\beta\eta} N$.
10. *Injectivity of Π .* If $\Gamma \vdash \Pi x:A.B = \Pi x:C.D : s$ then $\Gamma \vdash A = C : \text{type}$ and $\Gamma, x:A \vdash B = D : s$.

11. *Subject Reduction.* If $\Gamma \vdash M : A$ and $M \rightarrow_{\beta\eta} N$ then $\Gamma \vdash M = N : A$.
12. *Strong Normalization.* $\Gamma \vdash M : A$ implies M is strongly normalizing under $\rightarrow_{\beta\eta}$.
13. *Strengthening.* If $\Gamma, x:A, \Gamma' \vdash J$ and $x \notin \text{FV}(\Gamma') \cup \text{FV}(J)$ then $\Gamma, \Gamma' \vdash J$.

Lemma 2. If $\Gamma \vdash M : A$, $\Gamma \vdash N : B$ and $M \downarrow_{\beta\eta} N$ then there is an s such that $\Gamma \vdash M = N : A$ and $\Gamma \vdash A = B : s$.

Proof. By Subject Reduction, Splitting, Uniqueness of Types and equational reasoning.

We also observe without proof that Ext is equivalent to the following rule:

$$(\eta) \quad \frac{\Gamma \vdash M : \Pi x:A_1.A_2}{\Gamma \vdash \lambda x:A_1.M(x) = M : \Pi x:A_1.A_2}$$

3 Termination of Coquand's Algorithm

In this section we study properties of Coquand's algorithm, adapted to our presentation of the Logical Framework. This algorithm is based only on the syntax of the terms being compared, and contains no type information.

3.1 Definition

Coquand's algorithm is defined inductively by the inference rules in Figure 2. The algorithm $M \Longleftrightarrow N$ simply reduces its arguments M and N to weak-head normal form. The algorithm $M \longleftrightarrow N$ compares terms in weak-head normal form: the interesting cases are the non-structural rules λ -Left and λ -Right, where

$$\begin{array}{c}
 (\text{WHRed}) \quad \frac{P \longleftrightarrow Q}{M \Longleftrightarrow N} \quad (M \multimap_w P \text{ and } N \multimap_w Q) \\
 (\text{Var}) \quad x \longleftrightarrow x \quad (\text{type}) \quad \text{type} \longleftrightarrow \text{type} \\
 (\Pi) \quad \frac{A_1 \longleftrightarrow B_1 \quad A_2 \longleftrightarrow B_2}{\Pi x:A_1.A_2 \longleftrightarrow \Pi x:B_1.B_2} \\
 (\text{App}) \quad \frac{M_1 \longleftrightarrow N_1 \quad M_2 \longleftrightarrow N_2}{M_1(M_2) \longleftrightarrow N_1(N_2)} \quad (M_1 \text{ and } N_1 \text{ weak-head normal and not canonical}) \\
 (\lambda) \quad \frac{M \longleftrightarrow N}{\lambda x:A.M \longleftrightarrow \lambda x:B.N} \\
 (\lambda\text{-Left}) \quad \frac{M \longleftrightarrow N(x)}{\lambda x:A.M \longleftrightarrow N} \quad (N \text{ weak-head normal and not canonical}) \\
 (\lambda\text{-Right}) \quad \frac{M(x) \longleftrightarrow N}{M \longleftrightarrow \lambda x:B.N} \quad (M \text{ weak-head normal and not canonical})
 \end{array}$$

Fig. 2. Untyped Algorithm for Conversion for the Logical Framework

the left- or right-hand side is an abstraction and two terms are equivalent after an application of Ext.

We assume implicitly that an implementation of the algorithm will examine combinations of terms and evaluate the premisses given by the inference rules recursively. Axioms in the inference rules will return true, while combinations that do not appear in the inference rules will return false. Hence, the inference rules of the algorithm give us both an inductively defined relation and an algorithm yielding either true or false; clearly, the algorithm yields true iff there is a derivation using the inference rules.

Furthermore, observe that the inference rules are syntax-directed, meaning that at most one rule will apply for any pair of terms. This fact is used implicitly in the proofs below.

3.2 Termination and Completeness of Coquand's Algorithm

We now show that Coquand's algorithm terminates.

We begin by defining a measure where λ -abstractions $\lambda x:A.M$ are larger than applications to a variable $M(x)$. We use this measure as the base of the induction to show termination of the algorithm.

Definition 2. *Define the length of a normal term M recursively on its structure:*

$$\begin{array}{lll} |s| \equiv 1 & |\Pi x:A_1.A_2| \equiv |A_1| + |A_2| + 1 & |M(N)| \equiv |M| + |N| + 1 \\ |x| \equiv 1 & |\lambda x:A.M| \equiv |M| + 3 & \end{array}$$

Lemma 3 (Termination). *If M and N are β -normalizing then $M \iff N$ terminates.*

Proof. By nested induction on the sum of $|M^{nf}|$ and $|N^{nf}|$ and the sum of the lengths of the β -reduction sequences for M and N .

By Lemma 1 Case 1 M and N are weak-head normal or have weak-head reducts. If M or N has a weak-head reduct, then by WHRed $M \iff N$ terminates if $M^{wnf} \iff N^{wnf}$ terminates, where the latter follows by the induction hypothesis for reduction; $M \iff N$ terminates with the same result as $M^{wnf} \iff N^{wnf}$. Otherwise, M and N are in weak-head normal form. We perform case analysis on M and N to show that $M \iff N$ terminates; then $M \iff N$ terminates with the same result. We consider several cases:

- M is basic and N is an application. Then $M \iff N$ terminates in failure.
- $M \equiv M_1(M_2)$ and $N \equiv N_1(N_2)$. If M and N are weak-head normal then M_1 and N_1 must be weak-head normal and not abstractions. If M_1 or N_1 is a product then it is canonical, so $M_1(M_2) \iff N_1(N_2)$ fails immediately. Otherwise, $M^{nf} \equiv M_1^{nf}(M_2^{nf})$ and $N^{nf} \equiv N_1^{nf}(N_2^{nf})$, so by induction hypothesis $M_1 \iff N_1$ and $M_2 \iff N_2$ terminate. If both succeed then $M \iff N$ succeeds, and otherwise it fails.

- $M \equiv \lambda x:A_1.M_0$ and N is not canonical. Only rule λ -Left applies, and so $\lambda x:A_1.M_0 \longleftrightarrow N$ terminates if $M_0 \longleftrightarrow N(x)$ terminates. Clearly $M^{nf} \equiv \lambda x:A_1^{nf}.M_0^{nf}$ and $(N(x))^{nf} \equiv N^{nf}(x)$, and $|\lambda x:A_1^{nf}.M_0^{nf}| + |N^{nf}| = |M_0^{nf}| + 3 + |N^{nf}| > |M_0^{nf}| + |N^{nf}| + 2 = |M_0^{nf}| + |N^{nf}(x)|$, so $M_0 \longleftrightarrow N(x)$ terminates by induction hypothesis; if this succeeds then $\lambda x:A_1.M_0 \longleftrightarrow N$ succeeds, and otherwise it fails.
- $M \equiv \lambda x:A_1.M_0$ and $N \equiv \Pi y:B_1.B_2$. Then $M \longleftrightarrow N$ fails immediately.

Lemma 4 (Completeness). *If $\Gamma \vdash M = N : A$ then $M \longleftrightarrow N$.*

Proof. By Church–Rosser and Splitting it suffices to show that if $M \downarrow_{\beta\eta} N$ and $\Gamma \vdash M : A$ and $\Gamma \vdash N : B$ then $M \longleftrightarrow N$, which we show using the same induction principle as for Lemma 3. As in that lemma, we can distinguish two cases, depending on whether M and N are both weak-head normal. If either is not, then by Church–Rosser if $M \downarrow_{\beta\eta} N$ then $M^{wnf} \downarrow_{\beta\eta} N^{wnf}$, since M and N are well-typed, and by Subject Reduction $\Gamma \vdash M^{wnf} : A$ and $\Gamma \vdash N^{wnf} : B$. Therefore $M^{wnf} \longleftrightarrow N^{wnf}$ follows by the induction hypothesis on reduction sequences, so $M^{wnf} \longleftrightarrow N^{wnf}$ by inversion, and so $M \longleftrightarrow N$ by WHRed. We now consider the cases where M and N are in weak-head normal form: we show that if $M \downarrow_{\beta\eta} N$ then $M \longleftrightarrow N$, from which $M \longleftrightarrow N$. We consider several cases where M and N are in weak-head normal form.

- M and N are basic. If $x \downarrow_{\beta\eta} y$ then $x = y$, so $x \longleftrightarrow y$. type $\longleftrightarrow$ type.
- M is basic and N is an application. $x \downarrow_{\beta\eta} N_1(N_2)$ and type $\downarrow_{\beta\eta} N_1(N_2)$ are impossible.
- $M \equiv \lambda x:A_1.M_0$ and N not canonical. Let $\lambda x:A_1.M_0 \downarrow_{\beta\eta} N$, $\Gamma \vdash \lambda x:A_1.M_0 : A$ and $\Gamma \vdash N : B$. Then $M_0 \twoheadrightarrow M'_0(x)$ with $x \notin \text{FV}(M'_0)$, and $N \twoheadrightarrow M'_0$; hence $M_0 \downarrow_{\beta\eta} N(x)$, since N is weak-head normal and not canonical. By Generation $\Gamma, x:A_1 \vdash M_0 : A_2$ and $\Gamma \vdash \Pi x:A_1.A_2 = A : s$; by Splitting $\Gamma \vdash \Pi x:A_1.A_2 : s$, and by Generation $\Gamma \vdash A_1 : \text{type}$. Then by Lemma 2 $\Gamma \vdash A = B : s$, so $\Gamma \vdash N : \Pi x:A_1.A_2$; by Thinning $\Gamma, x:A_1 \vdash N : \Pi x : A_1.A_2$ and by App $\Gamma, x:A_1 \vdash N(x) : A_2$. Therefore, by induction hypothesis $M_0 \longleftrightarrow N(x)$ implies $\lambda x:A_1.M_0 \longleftrightarrow N$.
- $M \equiv \lambda x:A_1.M_0$ and $N \equiv \Pi y:B_1.B_2$. Suppose $\lambda x:A_1.M_0 \downarrow_{\beta\eta} \Pi y:B_1.B_2$, $\Gamma \vdash \lambda x:A_1.M_0 : A$ and $\Gamma \vdash \Pi y:B_1.B_2 : B$. Then $\Gamma \vdash \lambda x:A_1.M_0 : \Pi x:A_1.A_2$ and $\Gamma \vdash \Pi y:B_1.B_2 : s$ by Generation. By Lemma 2 $\Gamma \vdash \Pi x:A_1.A_2 = s : s'$, and by Church–Rosser $\Pi x:A_1.A_2 \downarrow_{\beta\eta} s$, which is impossible.

3.3 Correctness of the Algorithm

Our proof of correctness of Coquand’s algorithm is similar to his original proof, but we restate the proof because we rely on the metatheory of $\beta\eta$ -reduction rather than his logical relation over the algorithm.

Lemma 5 (Correctness).

- *If $\Gamma \vdash M, N : A$ and $M \longleftrightarrow N$ then $\Gamma \vdash M = N : A$.*

- If $M \longleftrightarrow N$, $\Gamma \vdash M : A$, $\Gamma \vdash N : B$ and M and N are not canonical then $\Gamma \vdash M = N : A$ and $\Gamma \vdash A = B : s$. If $M \longleftrightarrow N$ and $\Gamma \vdash M, N : A$ then $\Gamma \vdash M = N : A$.

Proof. By induction on the derivations of $M \longleftrightarrow N$ and $M \longleftrightarrow N$. We consider several cases:

- WHRed. By Subject Reduction $\Gamma \vdash M = P : A$ and $\Gamma \vdash N = Q : A$, by Splitting $\Gamma \vdash P, Q : A$, and by induction hypothesis $\Gamma \vdash P = Q : A$, so $\Gamma \vdash M = N : A$ by Symmetry and Transitivity.
- II. By Generation $\Gamma \vdash A_1, B_1 : \text{type}$, $\Gamma, x:A_1 \vdash A_2 : s$ and $\Gamma, x:B_1 \vdash B_2 : s$; by induction hypothesis $\Gamma \vdash A_1 = B_1 : \text{type}$, so by Context Replacement $\Gamma, x:A_1 \vdash B_2 : s$. By induction hypothesis again $\Gamma, x:A_1 \vdash A_2 = B_2 : s$.
- λ -Left. By assumption $\Gamma \vdash \lambda x:A_1.M_0 : C$, $\Gamma \vdash N : C$ and $M_0 \longleftrightarrow N(x)$. By Generation $\Gamma, x:A_1 \vdash M_0 : A_2$ and $\Gamma \vdash \Pi x:A_1.A_2 = C : s$. Therefore $\Gamma \vdash N : \Pi x:A_1.A_2$ by Sym and Eq and $\Gamma, x:A_1 \vdash N(x) : A_2$ by Weakening and App, and so by induction hypothesis $\Gamma, x:A_1 \vdash M_0 = N(x) : A_2$. Hence $\Gamma \vdash \lambda x:A_1.M_0 = N : \Pi x:A_1.A_2 = C$ by Ext.

4 Termination of Harper and Pfenning's Algorithm

Since it relies purely on the structure of terms, Coquand's algorithm cannot be used for type theories where equality may identify terms with different head variables, such as the extensional equalities on the unit type or singleton types. Such types can be important in applications, such as modules systems [1, 10], and Harper and Pfenning introduce type information into their algorithm in order to capture these types.

In this section we establish the decidability, completeness and correctness of Harper and Pfenning's algorithm for the Logical Framework.

4.1 Definition

We begin by defining a slight variant of Harper and Pfenning's algorithm.

The algorithm relies on an erasure function from the dependent types and kinds of the Logical Framework into simple types. We define our erasure into simple types formed only with constructors o and $\tau_1 \rightarrow \tau_2$, where Harper and Pfenning distinguish between sorts and constants in the Logical Framework; our approach should allow different judgements to be handled uniformly as in PTS. We use a single base type because we have used the same syntactic category for types and kinds, and it is more uniform not to distinguish between the two in the algorithm.

Formally, we define the simple types and contexts with the following BNF grammar:

$$\begin{aligned} \sigma, \tau \in S &::= o \mid \sigma \rightarrow \tau \\ \Delta \in X &::= () \mid \Delta, x:\sigma \end{aligned}$$

The erasure is defined inductively on the structure of types and kinds in weak-head normal form as follows:

$$\begin{array}{lll} \text{type}^- \equiv o & x^- \equiv o & (A(M))^- \equiv o \\ \text{kind}^- \equiv o & (\lambda x:A.M)^- \equiv o & (\Pi x:A_1.A_2)^- \equiv A_1^- \rightarrow A_2^- \end{array}$$

The definition extends in the obvious way to contexts. Erasure has the following simple properties, shown by induction on derivations.

Lemma 6. *If $\Gamma \vdash A : s$ then $([N/x]A)^- = A^-$. If $\Gamma \vdash A = B : s$ then $A^- = B^-$.*

The algorithm is defined inductively by the inference rules in Figure 3. Like Coquand's algorithm, this algorithm has a judgement $\Delta \vdash M \Longleftrightarrow N : \tau$ comparing arbitrary terms and a judgement $\Delta \vdash M \longleftrightarrow N : \tau$ comparing weak-head normal forms, but unlike Coquand's algorithm weak-head normalization in $\Longleftrightarrow$ is only performed at the base type, and terms at higher type are applied to variables and compared in the result type.

$$\begin{array}{l} \text{(Base)} \quad \frac{\Delta \vdash P \longleftrightarrow Q : o}{\Delta \vdash M \Longleftrightarrow N : o} \quad (M \rightarrow_w P \text{ and } N \rightarrow_w Q) \\ \text{(\(\rightarrow\))} \quad \frac{\Delta, x:\tau_1 \vdash M(x) \Longleftrightarrow N(x) : \tau_2}{\Delta \vdash M \Longleftrightarrow N : \tau_1 \rightarrow \tau_2} \quad (x \notin \text{dom}(\Delta)) \\ \text{(Var)} \quad \Delta \vdash x \longleftrightarrow x : \Delta(x) \quad \text{(type)} \quad \Delta \vdash \text{type} \longleftrightarrow \text{type} : o \\ \text{(\(\Pi\))} \quad \frac{\Delta \vdash A_1 \Longleftrightarrow B_1 : o \quad \Delta, x:A_1^- \vdash A_2 \Longleftrightarrow B_2 : o}{\Delta \vdash \Pi x:A_1.A_2 \longleftrightarrow \Pi x:B_1.B_2 : o} \\ \text{(App)} \quad \frac{\Delta \vdash M_1 \longleftrightarrow N_1 : \tau_1 \rightarrow \tau_2 \quad \Delta \vdash M_2 \Longleftrightarrow N_2 : \tau_1}{\Delta \vdash M_1(M_2) \longleftrightarrow N_1(N_2) : \tau_2} \end{array}$$

Fig. 3. Typed Algorithm for Conversion for the Logical Framework

We observe that, like Coquand's algorithm, this algorithm is syntax-directed: for $\Longleftrightarrow$ the context and type are part of the input and the algorithm returns true or false, and for $\longleftrightarrow$ the context is an input and the type is an output.

4.2 Termination and Completeness of the Algorithm

In this section we show the termination and completeness of Harper and Pfenning's type-directed algorithm for the Logical Framework.

For the following lemma, it is convenient to reason over traces of the algorithm itself, rather than the inference rules of those terms successfully related by the algorithm: we capture both success and failure of the implementation of the algorithm simultaneously. To this end, we shall write $\Delta; M; N; \tau \Rightarrow b$, for $b \in \{\text{tt}, \text{ff}\}$, to denote a trace of the algorithm for $\Delta \vdash M \Longleftrightarrow N : \tau$ yielding b as its result. Similarly, we shall write $\Delta; M; N \rightarrow v$, with $v \in S \cup \{\perp\}$, where $\Delta; M; N \rightarrow \tau$ if $\Delta \vdash M \longleftrightarrow N : \tau$ and $\Delta; M; N \rightarrow \perp$ if $\Delta \vdash M \longleftrightarrow N : \tau$ fails.

Lemma 7.

- If $\Delta, \Delta'; M; N; \tau \Rightarrow b$ and $x \notin \text{dom}(\Delta, \Delta')$ then $\Delta, x:\sigma, \Delta'; M; N; \tau \Rightarrow b$.
- If $\Delta, \Delta'; M; N \rightarrow v$ and $x \notin \text{dom}(\Delta, \Delta')$ then $\Delta, x:\sigma, \Delta'; M; N \rightarrow v$.

The following lemmas are by induction on types.

Lemma 8. *Let M and N be weak-head normal and not canonical. Then if $\Delta; M; N \rightarrow \tau$ then $\Delta; M; N; \tau \Rightarrow \text{tt}$; if $\Delta; M; N \rightarrow \perp$ then $\Delta; M; N; \tau \Rightarrow \text{ff}$ for any τ , and if $\Delta; M; N \rightarrow \tau$ then $\Delta; M; N; \tau' \Rightarrow b$ for any τ' .*

Lemma 9. *If $M \rightarrow_w P$ then $\Delta \vdash M \iff N : \tau$ terminates iff $\Delta \vdash P \iff N : \tau$ terminates, and with the same result, and symmetrically.*

We now prove the main results of this section.

Lemma 10 (Termination). *Suppose that M and N are β -normalizing. Then $\Delta \vdash M \iff N : \tau$ is terminating for any Δ and τ , and if M and N are weak-head normal and not canonical then $\Delta \vdash M \iff N : \tau$ is terminating for any Δ .*

Proof. We prove this by nested induction on the sum of $|M^{nf}|$ and $|N^{nf}|$ and the sum of the lengths of the β -reduction sequences of M and N . As for Coquand's algorithm, we use Lemma 1 Case 1 to perform case analysis on whether M and N are weak-head normal or not.

We consider several cases where M and N are weak-head normal.

- M and N basic. If $M = N = x$ and $x:\tau \in \Delta$ then $\Delta \vdash x \iff x : \tau$ succeeds. If $M = N = \text{type}$ then $\Delta \vdash \text{type} \iff \text{type} : o$ succeeds, and $\Delta \vdash \text{type} \iff \text{type} : \tau_1 \rightarrow \tau_2$ fails. Similarly, if $M \neq N$ or $M \equiv x \notin \text{dom}(\Delta)$ then $\Delta \vdash M \iff N : \tau$ fails. Each result lifts to $\iff$ by Lemma 8.
- $M \equiv \lambda x:A_1.M_0$ and N is not canonical. $\Delta \vdash M \iff N : o$ fails because no rules match $\Delta \vdash \lambda x:A_1.M_0 \iff N : o$. $\Delta \vdash \lambda x:A_1.M_0 \iff N : \tau_1 \rightarrow \tau_2$ terminates by definition iff $\Delta, x:\tau_1 \vdash (\lambda x:A_1.M_0)(x) \iff N(x) : \tau_2$, which by Lemma 9 terminates iff $\Delta, x:\tau_1 \vdash M_0 \iff N(x) : \tau_2$ terminates. But $\Delta, x:\tau_1 \vdash M_0 \iff N(x) : \tau_2$ terminates by induction hypothesis, since $|\lambda x:A_1^{nf}.M_0^{nf}| + |N^{nf}| = |M_0^{nf}| + 3 + |N^{nf}| > |M_0^{nf}| + |N^{nf}| + 2 = |M_0^{nf}| + |N^{nf}(x)|$.

Lemma 11 (Completeness). *If $\Gamma \vdash M = N : A$ then $\Gamma^- \vdash M \iff N : A^-$.*

Proof. By Church–Rosser and Splitting it suffices to show that if $M \downarrow_{\beta\eta} N$ and $\Gamma \vdash M : A$, $\Gamma' \vdash N : B$, $A^- = B^-$ and $\Gamma^- = \Gamma'^-$ then $\Gamma^- \vdash M \iff N : A^-$; and if $M \downarrow_{\beta\eta} N$ with M and N weak-head normal and not canonical, and $\Gamma \vdash M : A$, $\Gamma' \vdash N : B$ and $\Gamma^- = \Gamma'^-$, then $A^- = B^-$ and $\Gamma^- \vdash M \iff N : A^-$. We use the same induction principle as in Lemma 10; we consider several cases where M and N are weak-head normal.

- M and N basic. Clearly if $M \downarrow_{\beta\eta} N$ then $M = N = x$ or $M = N = \text{type}$. If $M = x$ then $\Gamma \vdash x : A$ and $\Gamma' \vdash x : B$ imply $\Gamma \vdash \Gamma(x) = A : \text{type}$, $\Gamma' \vdash \Gamma'(x) = B : \text{type}$, and $\Gamma^-(x) \in \Gamma^-$, so $\Gamma^- \vdash x \longleftrightarrow x : \Gamma(x)$ and $\Gamma^- \vdash x \iff x : \Gamma(x)$ as above. Also, $\Gamma^- = \Gamma'^-$ implies $\Gamma(x)^- = \Gamma'(x)$ implies $A^- = B^-$ by Lemma 6. If $M = \text{type}$ then $\Gamma \vdash \text{type} : A$ and $\Gamma' \vdash \text{type} : B$ imply $A = B = \text{kind}$, and $\Gamma^- \vdash \text{type} \longleftrightarrow \text{type} : o$ and $\Gamma^- \vdash \text{type} \iff \text{type} : o$.
- $M \equiv \lambda x:A_1.M_0$ and N not canonical. Let $\lambda x:A_1.M_0 \downarrow_{\beta\eta} N$, $\Gamma \vdash \lambda x:A_1.M_0 : A$, $\Gamma \vdash N : B$, and $A^- = B^-$. Then $M_0 \rightarrow_{\beta\eta} M'_0(x)$ with $x \notin \text{FV}(M'_0)$, and $N \rightarrow_{\beta\eta} M'_0$, and by Generation $\Gamma, x:A_1 \vdash M_0 : A_2$, $\Gamma \vdash \Pi x:A_1.A_2 = A$ and so $B \equiv \Pi x:B_1.B_2$ since $(\Pi x:A_1.A_2)^- = B^-$. Hence $M_0 \downarrow_{\beta\eta} N(x)$, since N is weak-head normal and not canonical, and $\Gamma, x:B_1 \vdash N(x) : B_2$ by Weakening and App, and $A_1^- = B_1^-$ implies $\Gamma^-, x:A_1^- = \Gamma'^-, x:B_1^-$, so $\Gamma^-, x:A_1^- \vdash M_0 \iff N(x) : A_2^-$ implies $\Gamma^-, x:A_1^- \vdash (\lambda x:A_1.M_0)(x) \iff N(x) : A_2^-$ by induction hypothesis implies $\Gamma^- \vdash \lambda x:A_1.M_0 \iff N : A_1^- \rightarrow A_2^- = (\Pi x:A_1.A_2)^-$.

4.3 Correctness of the Algorithm

The outline of our proof of the correctness of the algorithm follows Harper and Pfenning's proof. The primary difference is that because we rely on established metatheoretic results, Subject Reduction also applies to β -reducts at the level of types by assumption.

Lemma 12 (Correctness).

- If $\Gamma^- \vdash M \iff N : A^-$ and $\Gamma \vdash M, N : A$ then $\Gamma \vdash M = N : A$.
- If $\Gamma^- \vdash M \longleftrightarrow N : \tau$, $\Gamma \vdash M : A$, $\Gamma \vdash N : B$ and M and N not canonical then $\Gamma \vdash M = N : A$ and either $\Gamma \vdash A = B : s$ with $A^- = B^- = \tau$ or $A = B = \text{kind}$. If $\Gamma^- \vdash M \longleftrightarrow N : A^-$ and $\Gamma \vdash M, N : A$ then $\Gamma \vdash M = N : A$.

Proof. By induction on derivations. We consider several cases:

- Base. By Subject Reduction $\Gamma \vdash M = P : A$ and $\Gamma \vdash N = Q : A$, and by Splitting $\Gamma \vdash P, Q : A$, so by induction hypothesis $\Gamma \vdash P = Q : A$. Hence $\Gamma \vdash M = N : A$.
- Π . By assumption $\Gamma \vdash \Pi x:A_1.A_2, \Pi x:B_1.B_2 : C$, so $\Gamma \vdash A_1, B_1 : \text{type}$ by Generation, $\Gamma, x:A_1 \vdash A_2 : s$ and $\Gamma, x:B_1 \vdash B_2 : s'$, with derivations of $\Gamma \vdash C = \text{type} : \text{kind}$ or $C = \text{kind}$ from each derivation. Therefore $s = s'$, and so by induction hypothesis $\Gamma \vdash A_1 = B_1 : \text{type}$ and $\Gamma, x:A_1 \vdash A_2 = B_2 : s$, so $\Gamma \vdash \Pi x:A_1.A_2 = \Pi x:B_1.B_2 : s$.

5 System F

In this final technical section we show that our technique also works for a typed conversion algorithm for System F, hence extending our results beyond that of Harper and Pfenning.

5.1 Syntax

We begin by introducing the term syntax and inference rules for System F.

The following grammar presents the contexts, types, and terms of System F:

$$\begin{aligned} \Gamma \in C &::= () \mid \Gamma, x:A \\ A, B, C \in Y &::= X \mid A \rightarrow B \mid \forall X.A \\ M, N, P, Q \in T &::= x \mid \lambda x:A.M \mid M(N) \mid \Lambda X.M \mid M(A) \end{aligned}$$

Similar to the Logical Framework, we say that a term is canonical if it is of the form $\lambda x:A.M$ or $\Lambda X.M$.

We use the same notations for reduction, substitution and so on as for the Logical Framework. We say that a context Γ is valid if each $x \in \text{dom}(\Gamma)$ occurs exactly once in Γ . We write $\text{FTV}(A)$ for the free type variables occurring in A , and similarly for contexts.

Reduction is extended with β and η reductions for the type-level constructors:

$$\begin{aligned} (\Lambda X.M)(A) &\beta [A/X]M \\ \Lambda X.(M(X)) &\eta M \quad (X \notin \text{FTV}(M)) \end{aligned}$$

Weak-head reduction is similarly extended. The definitions of normal and weak-head normal are also extended in the natural way; observe that $(\Lambda X.M)(N)$ and $(\lambda x:A.M)(B)$ are normal and weak-head normal. Finally, the results of Lemma 1 extend to System F.

Our presentation of System F has only two judgements, $\Gamma \vdash M : A$ and $\Gamma \vdash M = N : A$; the inference rules for $\Gamma \vdash M : A$ are as follows:

$$\begin{aligned} (\text{Var}) \quad & \frac{\Gamma \text{ valid}}{\Gamma \vdash x : A} \quad (x:A \in \Gamma) \\ (\lambda) \quad & \frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash \lambda x:A.M : A \rightarrow B} \quad (\text{App}) \quad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash M(N) : B} \\ (\Lambda) \quad & \frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda X.M : \forall X.A} \quad (X \notin \text{FTV}(\Gamma)) \quad (\text{TyApp}) \quad \frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash M(B) : [B/X]A} \end{aligned}$$

The equality judgement is the evident typed extension of rules β and Ext for terms and types, as in Section 2.2.

System F enjoys a list of properties similar to those of Section 2.4, including Subject Reduction, Church–Rosser, Splitting, Uniqueness of Types, and so on. Due to a lack of space, we omit the full statement of these properties.

5.2 The Algorithm

The algorithm is defined by the inference rules in Figure 4.

5.3 Termination and Completeness of the Algorithm

The arguments for termination and completeness of the algorithm are very similar to the arguments for the Logical Framework, although they are simpler due to the lack of dependent types. We briefly outline the proofs here.

$$\begin{array}{c}
(\text{TyVar}) \quad \frac{\Gamma \vdash P \longleftrightarrow Q : X}{\Gamma \vdash M \longleftrightarrow N : X} \quad (M \dashv\!\!\!\rightarrow_w P \text{ and } N \dashv\!\!\!\rightarrow_w Q) \\
(\rightarrow) \quad \frac{\Gamma, x:A \vdash M(x) \longleftrightarrow N(x) : B}{\Gamma \vdash M \longleftrightarrow N : A \rightarrow B} \quad (x \notin \text{dom}(\Gamma)) \\
(\forall) \quad \frac{\Gamma \vdash M(X) \longleftrightarrow N(X) : A}{\Gamma \vdash M \longleftrightarrow N : \forall X.A} \quad (X \notin \text{FTV}(\Gamma) \cup \text{FTV}(M) \cup \text{FTV}(N)) \\
(\text{Var}) \quad \Gamma \vdash x \longleftrightarrow x : \Gamma(x) \\
(\text{App}) \quad \frac{\Gamma \vdash M_1 \longleftrightarrow N_1 : A \rightarrow B \quad \Gamma \vdash M_2 \longleftrightarrow N_2 : A}{\Gamma \vdash M_1(M_2) \longleftrightarrow N_1(N_2) : B} \\
(\text{TyApp}) \quad \frac{\Gamma \vdash M \longleftrightarrow N : \forall X.A}{\Gamma \vdash M(B) \longleftrightarrow N(B) : [B/X]A}
\end{array}$$

Fig. 4. Algorithm for Conversion for System F

We define the length function $|M|$ in the obvious way for System F, and $\Gamma; M; N; A \Rightarrow b$ and $\Gamma; M; N \rightarrow v$ are also extended.

Lemma 13. *Let M and N be weak-head normal and not canonical. Then if $\Gamma; M; N \rightarrow A$ then $\Gamma; M; N; A \Rightarrow \text{tt}$; if $\Gamma; M; N \rightarrow \perp$ then $\Gamma; M; N; A \Rightarrow \text{ff}$ for any A , and if $\Gamma; M; N \rightarrow A$ then $\Gamma; M; N; B \Rightarrow b$ for any B .*

Lemma 14. *If $M \rightarrow_w P$ then $\Gamma \vdash M \longleftrightarrow N : A$ terminates iff $\Gamma \vdash P \longleftrightarrow N : A$ terminates, and with the same result, and symmetrically.*

Lemma 15 (Termination). *If M and N are β -normalizing then $\Gamma \vdash M \longleftrightarrow N : A$ is terminating for any Γ and A ; if M and N are weak-head normal and not canonical then $\Gamma \vdash M \longleftrightarrow N : A$ is terminating for any Γ .*

Proof. By nested induction on the sum of $|M^{nf}|$ and $|N^{nf}|$ and the sum of the lengths of β -reduction sequences for M and N . As in the previous sections, if M and N are not weak-head normal then the result follows by the nested induction hypothesis.

We consider several cases where M and N are weak-head normal.

- $M \equiv \lambda X.M_0$ and N not canonical. $\Gamma \vdash \lambda X.M_0 \longleftrightarrow N : X$ fails immediately, and $\Gamma \vdash \lambda X.M_0 \longleftrightarrow N : A \rightarrow B$ fails since $\Gamma, x:A \vdash (\lambda X.M_0)(x) \longleftrightarrow N(x) : B$ fails by Lemma 13, since $\Gamma, x:A \vdash (\lambda X.M_0)(x) \longleftrightarrow N(x) : B$ fails. Suppose $A \equiv \forall X.B$; then $\Gamma \vdash M_0 \longleftrightarrow N(X) : B$ terminates by induction hypothesis on the combined length of the normal forms of $\lambda X.M_0$ and N , $\Gamma \vdash (\lambda X.M_0)(X) \longleftrightarrow N(X) : B$ terminates by Lemma 14, and so $\Gamma \vdash \lambda X.M_0 \longleftrightarrow N : \forall X.B$ terminates.
- $M \equiv M_1(C)$ with M_1 weak-head normal and not canonical, and $N \equiv N_1(D)$ with N_1 weak-head normal and not canonical. If $C \neq D$ then the algorithm fails. Otherwise, $\Gamma \vdash M_1 \longleftrightarrow N_1 : A$ terminates: if it fails or if $A \neq \forall X.B$ then $\Gamma \vdash M_1(C) \longleftrightarrow N_1(C)$ fails, and otherwise $\Gamma \vdash M_1(C) \longleftrightarrow N_1(C) : [C/X]B$ succeeds. The results lift to $\Gamma \vdash M_1(C) \longleftrightarrow N_1(C) : D$ by Lemma 13.

Lemma 16 (Completeness). *If $\Gamma \vdash M = N : A$ then $\Gamma \vdash M \iff N : A$.*

Proof. By Church–Rosser and Splitting it suffices to show that if $M \downarrow_{\beta\eta} N$ and $\Gamma \vdash M, N : A$ then $\Gamma \vdash M \iff N : A$, and if M and N are weak-head normal and not canonical then $\Gamma \vdash M \iff N : A$. We show this by the same induction used in Lemma 15; we consider several cases here.

- $M \equiv \lambda X.M_0$ and N not canonical. Suppose $\lambda X.M_0 \downarrow_{\beta\eta} N$ and $\Gamma \vdash \lambda X.M_0, N : A$. By inversion $\Gamma \vdash M_0 : B$ and $A \equiv \forall X.B$, and so $\Gamma \vdash (\lambda X.M_0)(X), N(X) : B$ by TyApp. Hence by induction hypothesis $\Gamma \vdash M_0 \iff N(X) : B$, so $\Gamma \vdash \lambda X.M_0 \iff N : \forall X.B$.
- $M \equiv M_1(C)$ with M_1 weak-head normal and not canonical, and $N \equiv N_1(D)$ with N_1 weak-head normal and not canonical. Suppose $\Gamma \vdash M_1(C), N_1(D) : A$. By inversion $\Gamma \vdash M_1 : \forall X.E$, $[C/X]E = A$, $\Gamma \vdash N_1 : \forall X.F$, and $[D/X]F = A$. Furthermore, $M_1 \rightarrow_w P_1$ and $N_1 \rightarrow_w P_1$ and $C = D$, so by Subject Reduction $\Gamma \vdash M_1 = P_1 : \forall X.E$ and $\Gamma \vdash N_1 = P_1 : \forall X.F$, so by Uniqueness of Types $\forall X.E = \forall X.F$. Therefore by induction hypothesis $\Gamma \vdash M_1 \iff N_1 : \forall X.E$, so $\Gamma \vdash M_1(C) \iff N_1(C) : [C/X]E$.

5.4 Correctness of the Algorithm

We now show that the algorithm is correct for System F.

Lemma 17 (Correctness).

- If $\Gamma \vdash M \iff N : A$ and $\Gamma \vdash M, N : A$ then $\Gamma \vdash M = N : A$.
- If $\Gamma \vdash M \iff N : A$, $\Gamma \vdash M : B$ and $\Gamma \vdash N : C$ then $\Gamma \vdash M = N : A$ and $A = B = C$.

Proof. By induction on derivations. We consider several cases:

- $\rightarrow$. We have $\Gamma \vdash M, N : A \rightarrow B$. By Weakening $\Gamma, x:A \vdash M, N : A \rightarrow B$, and so by Var and App $\Gamma, x:A \vdash M(x), N(x) : B$, and by induction hypothesis $\Gamma, x:A \vdash M(x) = N(x) : B$. By λ and Ext $\Gamma \vdash M = N : A \rightarrow B$.
- Var. By inversion $\Gamma \vdash x : B$ implies $B = \Gamma(x)$, and $\Gamma \vdash x = x : \Gamma(x)$.
- TyApp. We have $\Gamma \vdash M(B) : C$, $\Gamma \vdash N(B) : D$, and $\Gamma \vdash M(B) \iff N(B) : [B/X]A$. By inversion $\Gamma \vdash M : \forall X.E$ and $\Gamma \vdash N : \forall X.F$. By induction hypothesis $\Gamma \vdash M = N : \forall X.A$ with $\forall X.A = \forall X.E = \forall X.F$, so $A = E = F$ and $[B/X]A = [B/X]E = [B/X]F$. Hence $\Gamma \vdash M(B) = N(B) : [B/X]A$.

6 Conclusions and Future Work

We have demonstrated that the standard metatheory for the Logical Framework and System F for $\beta\eta$ -equality is sufficient to justify algorithms for conversion not immediately modeled by reduction. We used a simple inductive measure to show the completeness and decidability of the algorithms.

A natural extension of this work would be to study the algorithm for conversion for the Calculus of Constructions with $\beta\eta$ -equality. We have made substantial progress towards this goal by showing how type dependency can be erased and reconstructed for the Logical Framework, and how polymorphism can be justified. Existing developments using erasure to study metatheory of dependent type theories [5] suggest that the type-directed algorithm for the non-dependent version of a calculus could be used to typecheck the dependently typed version.

One of the primary motivations for Harper and Pfenning's algorithm was singleton types, where Coquand's untyped algorithm may fail to identify equal terms. It seems that it should be possible to extend our technique given the metatheory for $\beta\eta$ -equality, but one of the benefits of giving an algorithm directly is that it addresses problems with the reduction relation, such as failure of confluence. This is an interesting area for further research.

Acknowledgments

I would like to thank Bob Harper for stimulating my renewed interest in this topic, and the anonymous referees and Andreas Abel for helpful corrections and comments. I would also like to thank my wife Adriana Compagnoni for her encouragement and support as I was writing this article.

References

1. D. Aspinall. Subtyping with singleton types. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic*, pages 1–15. Springer, Berlin, 1994.
2. H. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbai, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1991.
3. T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*. Cambridge University Press, 1991.
4. H. Geuvers. *Logics and Type Systems*. PhD thesis, Katholieke Universiteit Nijmegen, Sept. 1993.
5. H. Geuvers and M.-J. Nederhof. A modular proof of strong normalization for the calculus of constructions. *Journal of Functional Programming*, 1(2):155–189, Apr. 1991.
6. H. Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, Aug. 1994.
7. H. Goguen. A syntactic approach to eta equality in type theory. In *Symposium on Principles of Programming Languages*, Jan. 2005.
8. R. Harper and F. Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Trans. on Computational Logic*, 2004. To appear.
9. A. Salvesen. The Church-Rosser property for pure type systems with $\beta\eta$ -reduction, Nov. 1991. Unpublished manuscript.
10. C. A. Stone and R. Harper. Equivalence and singletons. *ACM Transactions on Programming Languages and Systems*, 2004. Submitted.
11. J. Vanderwaart and K. Cray. A simplified account of the metatheory of linear LF. *Electronic Notes in Theoretical Computer Science*, 70(2), 2002. Extended version available as Technical Report CMU-CS-01-154.