A Simplified Account of the Metatheory of Linear LF

Joseph C. Vanderwaart and Karl Crary

April 17, 2002

CMU-CS-01-154

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a variant of the linear logical framework LLF that avoids the restriction that well-typed forms be in pre-canonical form and adds λ-abstraction at the level of families. We abandon the use of β-conversion as definitional equality in favor of a set of typed definitional equality judgments that include rules for parallel conversion and extensionality. We show type-checking is decidable by giving an algorithm to decide definitional equality for well-typed terms and showing the algorithm is sound and complete. The algorithm and the proof of its correctness are simplified by the fact that they apply only to well-typed terms and may therefore ignore the distinction between intuitionistic and linear hypotheses.

This material is based on work supported in part by NSF grants CCR-9984812 and CCR-0121633. Any opinions, findings, and conclusions or recommendations in this publication are those of the authors and do not reflect the views of this agency.

Keywords: Logical Frameworks, Type Theory, Linear Logic

1 Introduction

Decidability of type-checking and the existence of canonical forms for well-typed terms are arguably the two most important metatheoretic results for a logical framework such as LF [5]. Type-checking is essential because the checking of proofs reduces to type-checking of the terms that represent them; canonical forms are crucial because it is the canonical terms (of certain types) that may be proven via an "adequacy" theorem to be in a meaningful correspondence with propositions and proofs in a logic.

Canonical forms in the LF type theory are β-normal, η-long forms. It therefore seems reasonable to take definitional equality to be $\beta\eta$-conversion, and decide whether two terms are equal by reducing them to $\beta\eta$-normal form and comparing; unfortunately, η-reduction is not as well behaved as β-reduction and so this approach encounters significant problems. The original presentation of LF by Harper, Honsell and Plotkin [5] (hereafter "HHP") avoided the difficulties of η-reduction by using β-conversion as definitional equality even though this destroyed the property that every term is equal to some canonical form.

Felty's Canonical LF [4] is a version of LF where all well-typed objects and families are in canonical form, avoiding all issues of definitional equality. Felty showed that Canonical LF is essentially the same as full LF if typing derivations are restricted to pre-canonical terms (those whose β-normal forms are canonical) with β-conversion as definitional equality. A similar approach was taken by Cervesato and Pfenning for their presentation of the linear logical framework LLF [1] (hereafter "CP"). The typing rules of LLF forced well-formed terms into η-long form, making all well-typed β-normal forms canonical and rendering any η rules for definitional equality unnecessary.

Subsequent to the original definition of LLF, Harper and Pfenning [6, 7] (hereafter, "HP") gave an alternate formulation of ordinary LF that allowed a clean treatment of definitional equality without having to restrict terms to pre-canonical form. Their approach was based on the use of a typed definitional equality judgment as opposed to an untyped reduction relation—that is, they focused on comparing two objects *at a certain type* and *in a certain typing context*, rather than $\beta\eta$-normalizing them in isolation. The decidability of type-checking and the existence of canonical forms for well-typed terms were then established by giving a set of algorithmic judgments that are sound and complete with respect to definitional equality and can be instrumented to extract canonical forms from the terms they compare. The algorithm is similar to one introduced by Coquand [2], except that Coquand's algorithm performs η-expansion based on the shapes of terms, while HP's is directed by types and kinds. The type-directed nature of the algorithm has the advantage of making it scalable to theories such as LLF that contain unit types. Apart from the typed, declarative formulation of definitional equality, the LF type theory considered by HP was essentially the same as that of HHP, except that the λ-abstraction construct at the level of type families had to be removed for technical reasons. This was not considered a problem, as experience with LF had shown that this form of abstraction was not needed in practice.

In this paper, we present a variant of LLF that employs a typed formulation of definitional equality in the style of HP and extend HP's equality algorithm to handle the linear constructs of LLF. We also resolve the technical issues that necessitated the removal of the family-level λ-abstraction, and so we are able to give a variant of LLF that includes abstraction at the family level even though CP's LLF did not have this feature. This result has been applied by the authors to establish the decidability of typing in the LTT type theory for certified code [3], and has been extended by Polakow for the Ordered Linear Logical Framework [9].

1.1 Overview

The structure of this report is as follows. In Section 2 we present our variant of the Linear LF type theory, which will essentially be an extension of HP's formulation of LF with linear implication, additive conjunction and additive truth. We will also add a family-level λ-abstraction construct similar to the one that appeared in HHP. Our presentation of the theory will include the typed definitional equality rules we are using to replace the untyped conversion relation of CP. Section 3 will establish some elementary properties of the typing and definitional equality judgments, most notably the regularity, inversion and functionality lemmas that will play a major role in the rest of our discussion of the theory. Another property of the system, which we call injectivity, will be proved in Section 4. HP proved their analogue of injectivity by straightforward induction, but the family-level λ-abstraction construct we have added to the language makes this simple proof impossible; we will use a logical relation to prove our version.

With the fundamental properties of the type theory established, the remainder of the report focuses on proving that definitional equality is decidable. Section 5 defines a set of "algorithmic equality" judgments, in the style of HP, that are directed by the shapes of the types of the objects being compared. Some elementary properties of the algorithmic judgments are also proven in that section. Sections 6 and 7 prove that the new judgments are correct—that is, sound and complete with respect to definitional equality—for well-typed terms; soundness is established in Section 6 and completeness in Section 7. Both of these proofs more or less follow the method of HP, although the linear features of Linear LF make the soundness proof somewhat more complicated. Finally in Section 8 we verify that the "algorithmic" judgments we introduced in Section 5 do in fact define an algorithm—that is, that the search for an algorithmic equality derivation between two well-typed terms will always terminate. This result is equivalent to the decidability of definitional equality and thus implies the decidability of typing in Linear LF.

2 A Variant of Linear LF

In this section we will present our variant of Linear LF type theory. This version of Linear LF has essentially the same syntax as that presented by CP, with the addition of λ-abstraction at the level of families. The typing rules we give here, however, differ from CP's in that we do not restrict terms to pre-canonical form. Another minor difference is that we use two separate contexts in our typing judgments, one for intuitionistic and one for linear assumptions, rather than combining the two. Since none of the variables declared in the linear context will appear in any types, we may identify linear contexts that differ only in the order of declarations and thus we do not need a formal context-splitting judgment for the linear application rules. Our variant of Linear LF can also be thought of as a linear version of the LF type theory as presented by HP.

The syntax of Linear LF terms is given by the following grammar.

Kinds	K	$::=$	$\mathsf{Type} \mid \Pi u{:}A.K$
Families	A	$::=$	$a \mid \lambda u{:}A_1.A_2 \mid A\,M \mid$
			$\Pi u{:}A_1.A_2 \mid$
			$A_1 \multimap A_2 \mid A_1 \mathbin{\&} A_2 \mid \top$
Objects	M	$::=$	$u \mid c \mid$
			$\lambda u{:}A.M \mid M_1\,M_2 \mid$
			$\hat{\lambda} u{:}A.M \mid M_1\,\hat{}\,M_2 \mid \langle M_1, M_2\rangle \mid \pi_i M \mid \langle\rangle$
Contexts	Γ	$::=$	$\epsilon \mid \Gamma, a{:}K \mid \Gamma, u{:}A$
Linear Contexts	Δ	$::=$	$\epsilon \mid \Delta, u\hat{:}A$

There are three classes, or levels, of terms: objects, families and kinds. Kinds classify families: families of kind Type are called *types*, and classify objects, while $\Pi u{:}A.K$ is the (possibly dependent) kind of a function from objects of family A to families of kind K. At the family level we have constants (a), function abstractions and applications as well as (intuitionistic) dependent function types, linear implication ($\multimap$), additive conjunction ($\&$), and additive truth ($\top$). The object level contains variables (u), constants (c), intuitionistic λ-abstraction and application ($M_1\,M_2$), linear $\hat{\lambda}$-abstraction and application ($M_1\,\hat{}\,M_2$), pairs ($\langle M_1, M_2, \rangle$), projections ($\pi_i\,M$) and the object $\langle\rangle$ that inhabits $\top$.

Contexts, both intuitionistic (Γ) and linear (Δ), will be used in the typing and definitional equality rules. Note that the order of assumptions in an intuitionistic context may be significant, since term variables may appear in types. The typing rules will prevent types from depending on any linear assumptions, which justifies our grouping all the linear assumptions into a separate linear context. We regard linear contexts that differ only in the ordering of assumptions as identical.

If the variable u does not appear in the family B, then we may write $A \to B$ for $\Pi u{:}A.B$. Similarly, $A \to K$ will mean $\Pi u{:}A.K$ if u does not appear in K. As usual, we denote by $E[E'_1, \ldots, E'_n/u_1, \ldots, u_n]$ the simultaneous capture-avoiding substitution of the terms E'_1 through E'_n for the corresponding variables u_1 through u_n in E.

2.1 Typing Rules

Signatures and Contexts ($\vdash S$ sig, $\vdash \Gamma$ $\mathsf{context}$, $\Gamma \vdash \Delta$ $\mathsf{context}$)

In LF and LLF, the types and kinds of constants are given by a *signature*. It is by the choice of signature that the logical framework is instantiated to represent a particular logic. For simplicity, in the bulk of this paper we will tacitly assume a fixed signature S; the exception is in the well-formedness rules for signatures, where we use a subscripted turnstile ($\vdash_S$) to indicate that certain premises are to be understood with respect to the signature mentioned in the conclusion.

The validity judgment for linear contexts specifies an intuitionistic context giving types to all the free variables that may occur in the linear assumptions.

$$\frac{}{\vdash \epsilon\ \mathsf{sig}} \qquad \frac{\vdash S\ \mathsf{sig} \quad \vdash_S K : \mathsf{kind}}{\vdash S, a{:}K\ \mathsf{sig}}\ (a \notin \mathrm{Dom}(S)) \qquad \frac{\vdash S\ \mathsf{sig} \quad \vdash_S A : \mathsf{Type}}{\vdash S, c{:}A\ \mathsf{sig}}\ (c \notin \mathrm{Dom} S)$$

$$\frac{}{\vdash \epsilon\ \mathsf{context}} \qquad \frac{\vdash \Gamma\ \mathsf{context} \quad \Gamma \vdash A : \mathsf{Type}}{\vdash \Gamma, u{:}A\ \mathsf{context}}\ (u \notin \mathrm{Dom}(\Gamma))$$

$$\frac{}{\Gamma \vdash \epsilon\ \mathsf{context}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{context} \quad \Gamma \vdash A : \mathsf{Type}}{\Gamma \vdash \Delta, u\hat{:}A\ \mathsf{context}}\ (u \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Delta))$$

Kinds ($\Gamma \vdash K : \mathsf{kind}$)

The restriction that linear assumptions not appear in types or kinds is enforced by using only an intuitionistic context for kind and family judgments, and forcing the linear context in which an object is typed for dependent application to be empty.

$$\frac{}{\Gamma \vdash \mathsf{Type} : \mathsf{kind}} \qquad \frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma, u{:}A \vdash K : \mathsf{kind}}{\Gamma \vdash \Pi u{:}A.K : \mathsf{kind}}$$

Families ($\Gamma \vdash A : K$)

$$\frac{}{\Gamma \vdash a : K}\ (S(a) = K) \qquad \frac{\Gamma \vdash A : \Pi u{:}A_1.K \quad \Gamma; \epsilon \vdash M : A_1}{\Gamma \vdash A\,M : K[M/u]} \qquad \frac{\Gamma \vdash A_1 : \mathsf{Type} \quad \Gamma \vdash A_2 : \mathsf{Type}}{\Gamma \vdash A_1 \multimap A_2 : \mathsf{Type}}$$

$$\frac{\Gamma \vdash A_1 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash A_2 : K}{\Gamma \vdash \lambda u{:}A_1.A_2 : \Pi u{:}A_1.K} \qquad \frac{\Gamma \vdash A_1 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash A_2 : \mathsf{Type}}{\Gamma \vdash \Pi u{:}A_1.A_2 : \mathsf{Type}} \qquad \frac{\Gamma \vdash A_1 : \mathsf{Type} \quad \Gamma \vdash A_2 : \mathsf{Type}}{\Gamma \vdash A_1 \,\&\, A_2 : \mathsf{Type}}$$

$$\frac{}{\Gamma \vdash \top : \mathsf{Type}} \qquad \frac{\Gamma \vdash A : K' \quad \Gamma \vdash K' = K : \mathsf{kind}}{\Gamma \vdash A : K}$$

Objects ($\Gamma; \Delta \vdash M : A$)

$$\frac{}{\Gamma; \epsilon \vdash c : A}\ (S(c) = A) \qquad \frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma; \Delta, u\hat{:}A \vdash M : B}{\Gamma; \Delta \vdash \hat{\lambda} u{:}A.M : A \multimap B} \qquad \frac{}{\Gamma; \Delta \vdash \langle\rangle : \top}$$

$$\frac{}{\Gamma; \epsilon \vdash u : A}\ (\Gamma(u) = A) \qquad \frac{\Gamma; \Delta \vdash M_1 : A_1 \quad \Gamma; \Delta \vdash M_2 : A_2}{\Gamma; \Delta \vdash \langle M_1, M_2 \rangle : A_1 \& A_2} \qquad \frac{\Gamma; \Delta \vdash M : A' \quad \Gamma \vdash A' = A : \mathsf{Type}}{\Gamma; \Delta \vdash M : A}$$

$$\frac{}{\Gamma; u\hat{:}A \vdash u : A}$$

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma, u{:}A; \Delta \vdash M : B}{\Gamma; \Delta \vdash \lambda u{:}A.M : \Pi u{:}A.B} \qquad \frac{\Gamma; \Delta \vdash M : A_1 \& A_2}{\Gamma; \Delta \vdash \pi_i M : A_i}\ (i = 1, 2)$$

$$\frac{\Gamma;\Delta \vdash M_1 : \Pi u{:}A.B \quad \Gamma;\epsilon \vdash M_2 : A}{\Gamma;\Delta \vdash M_1\, M_2 : B[M_2/u]} \qquad \frac{\Gamma;\Delta_1 \vdash M_1 : A \multimap B \quad \Gamma;\Delta_2 \vdash M_2 : A}{\Gamma;\Delta_1,\Delta_2 \vdash M_1\,\hat{}\,M_2 : B}$$

2.2 Definitional Equality Rules

The definitional equality rules include compatibility rules for all the syntactic constructs of the calculus, as well as parallel conversion rules and extensionality rules for each of the type and kind constructors of the theory. Symmetry and transitivity of equality are explicitly included as rules; reflexivity is shown to be admissible.

Kinds ($\Gamma \vdash K = L : \mathsf{kind}$)

$$\frac{}{\Gamma \vdash \mathsf{Type} = \mathsf{Type} : \mathsf{kind}} \qquad \frac{\Gamma \vdash K_2 = K_1 : \mathsf{kind}}{\Gamma \vdash K_1 = K_2 : \mathsf{kind}} \qquad \frac{\Gamma \vdash K_1 = K_3 : \mathsf{kind} \quad \Gamma \vdash K_3 = K_2 : \mathsf{kind}}{\Gamma \vdash K_1 = K_2 : \mathsf{kind}}$$

$$\frac{\Gamma \vdash A_1 : \mathsf{Type} \quad \Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash K_1 = K_2 : \mathsf{kind}}{\Gamma \vdash \Pi u{:}A_1.K_1 = \Pi u{:}A_2 s.K_2 : \mathsf{kind}}$$

Families ($\Gamma \vdash A = B : K$)

$$\frac{}{\Gamma \vdash a = a : K}\ (S(a) = K) \qquad \frac{}{\Gamma \vdash \top = \top : \mathsf{Type}} \qquad \frac{\Gamma \vdash A_2 = A_1 : K}{\Gamma \vdash A_1 = A_2 : K}$$

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \multimap B_1 = A_2 \multimap B_2 : \mathsf{Type}} \qquad \frac{\Gamma \vdash A_1 = A_3 : K \quad \Gamma \vdash A_3 = A_2 : K}{\Gamma \vdash A_1 = A_2 : K}$$

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \& B_1 = A_2 \& B_2 : \mathsf{Type}} \qquad \frac{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K \quad \Gamma;\epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash A_1\, M_1 = A_2\, M_2 : K[M_1/u]}$$

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash \Pi u{:}A_1.B_1 = \Pi u{:}A_2.B_2 : \mathsf{Type}} \qquad \frac{\Gamma \vdash A_1 = A_2 : K' \quad \Gamma \vdash K' = K : \mathsf{kind}}{\Gamma \vdash A_1 = A_2 : K}$$

$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma, u{:}A \vdash B_1 = B_2 : K}{\Gamma \vdash \lambda u{:}A_1.B_1 = \lambda u{:}A_2.B_2 : \Pi u{:}A.K}$$

$$\frac{\Gamma \vdash B : \mathsf{Type} \quad \Gamma, u{:}B \vdash A_1 = A_2 : K \quad \Gamma;\epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash (\lambda u{:}B.A_1)\, M_1 = A_2[M_2/u] : K[M_1/u]}$$

$$\frac{\Gamma \vdash B : \mathsf{Type} \quad \Gamma \vdash A_1 : \Pi u{:}B.K \quad \Gamma \vdash A_2 : \Pi u{:}B.K \quad \Gamma, u{:}B \vdash A_1\, u = A_2\, u : K}{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K}$$

Objects ($\Gamma;\Delta \vdash M = N : A$)

Compatibility

$$\frac{}{\Gamma; \epsilon \vdash u = u : A}\ (\Gamma(u) = A) \qquad \frac{\Gamma; \Delta_1 \vdash M_1 = M_2 : A \multimap B \quad \Gamma; \Delta_2 \vdash N_1 = N_2 : A}{\Gamma; \Delta_1, \Delta_2 \vdash M_1\,\hat{}\,N_1 = M_2\,\hat{}\,N_2 : B}$$

$$\frac{}{\Gamma; \epsilon \vdash c = c : A}\ (S(c) = A) \qquad \frac{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \quad \Gamma; \Delta \vdash N_1 = N_2 : A_2}{\Gamma; \Delta \vdash \langle M_1, N_1 \rangle = \langle M_2, N_2 \rangle : A_1 \mathbin{\&} A_2}$$

$$\frac{}{\Gamma; u\hat{:}A \vdash u = u : A}$$

$$\frac{\Gamma; \Delta \vdash M_1 = M_2 : \Pi u{:}A.B \quad \Gamma; \epsilon \vdash N_1 = N_2 : A}{\Gamma; \Delta \vdash M_1\, N_1 = M_2\, N_2 : B[N_1/u]} \qquad \frac{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \mathbin{\&} A_2}{\Gamma; \Delta \vdash \pi_i M_1 = \pi_i M_2 : A_i}\ (i = 1, 2)$$

$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma, u{:}A; \Delta \vdash M_1 = M_2 : B}{\Gamma; \Delta \vdash \lambda u{:}A_1.M_1 = \lambda u{:}A_2.M_2 : \Pi u{:}A.B}$$

$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma; \Delta, u\hat{:}A \vdash M_1 = M_2 : B}{\Gamma; \Delta \vdash \hat{\lambda} u{:}A_1.M_1 = \hat{\lambda} u{:}A_2.M_2 : A \multimap B}$$

Type Conversion

$$\frac{\Gamma; \Delta \vdash M_1 = M_2 : A' \quad \Gamma \vdash A' = A : \mathsf{Type}}{\Gamma; \Delta \vdash M_1 = M_2 : A}$$

Equivalence

$$\frac{\Gamma; \Delta \vdash M_2 = M_1 : A}{\Gamma; \Delta \vdash M_1 = M_2 : A} \qquad \frac{\Gamma; \Delta \vdash M_1 = M_3 : A \quad \Gamma; \Delta \vdash M_3 = M_2 : A}{\Gamma; \Delta \vdash M_1 = M_2 : A}$$

Parallel Conversion

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma, u{:}A; \Delta \vdash M_1 = M_2 : B \quad \Gamma; \epsilon \vdash N_1 = N_2 : A}{\Gamma; \Delta \vdash (\lambda u{:}A.M_1)\, N_1 = M_2[N_2/u] : B[N_1/u]}$$

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma; \Delta_1, u\hat{:}A \vdash M_1 = M_2 : B \quad \Gamma; \Delta_2 \vdash N_1 = N_2 : A}{\Gamma; \Delta_1, \Delta_2 \vdash (\hat{\lambda} u{:}A.M_1)\,\hat{}\,N_1 = M_2[N_2/u] : B}$$

$$\frac{\Gamma; \Delta \vdash M_1 = N_1 : A_1 \quad \Gamma; \Delta \vdash M_2 = N_2 : A_2}{\Gamma; \Delta \vdash \pi_i \langle M_1, M_2 \rangle = N_i : A_i}\ (i = 1, 2)$$

Extensionality

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma; \Delta \vdash M_1 : \Pi u{:}A.B \quad \Gamma; \Delta \vdash M_2 : \Pi u{:}A.B \quad \Gamma, u{:}A; \Delta \vdash M_1\, u = M_2\, u : B}{\Gamma; \Delta \vdash M_1 = M_2 : \Pi u{:}A.B}$$

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma; \Delta \vdash M_1 : A \multimap B \quad \Gamma; \Delta \vdash M_2 : A \multimap B \quad \Gamma; \Delta, u\hat{:}A \vdash M_1\,\hat{}\,u = M_2\,\hat{}\,u : B}{\Gamma; \Delta \vdash M_1 = M_2 : A \multimap B}$$

$$\frac{\begin{array}{c}\Gamma; \Delta \vdash M_1 : A_1 \mathbin{\&} A_2 \quad \Gamma; \Delta \vdash M_2 : A_1 \mathbin{\&} A_2 \\ \Gamma; \Delta \vdash \pi_i M_1 = \pi_i M_2 : A_i \text{ for } i = 1, 2\end{array}}{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \mathbin{\&} A_2}$$

$$\frac{\Gamma; \Delta \vdash M : \top \quad \Gamma; \Delta \vdash N : \top}{\Gamma; \Delta \vdash M = N : \top}$$

2.3 Substitutions

In our logical relations proofs, we will make use of the concept of a *substitution*: that is, a finite mapping from variables (u) to object terms (M). As a matter of notation, we will use the letter γ to stand for a substitution if we intend that all variables appearing free in $\gamma(u)$ be intuitionistic, rather than linear. We will call such a substitution an *intuitionistic substitution*; for substitutions that may produce free linear variables we will use the letter σ.

If Γ is a context, then we will denote by id_Γ the identity substitution on $\mathrm{Dom}(\Gamma)$. If σ is a substitution, then $\sigma[u \mapsto M]$ will denote the substitution that maps u to M and maps v to $\sigma(v)$ for $v \neq u$.

If $\mathcal{O}$ is a kind, family or object term, then $\sigma\mathcal{O}$ will denote the result of replacing every free occurrence of every variable $u \in \mathrm{Dom}(\sigma)$ in $\mathcal{O}$ by $\sigma(u)$. These replacements are performed simultaneously, and bound variables are implicitly renamed to avoid capture. Note that variables not in $\mathrm{Dom}(\sigma)$ are left unchanged by this operation.

For intuitionistic substitutions, we lift the well-formedness and equality judgments on objects in the usual way, enforcing the non-occurrence of linear variables by keeping the linear context empty. Specifically, we define:

1. $\Gamma' \vdash \gamma : \Gamma$ iff $\mathrm{Dom}(\gamma) = \mathrm{Dom}(\Gamma)$ and for every $u \in \mathrm{Dom}(\Gamma)$, $\Gamma'; \epsilon \vdash \gamma(u) : \gamma(\Gamma(u))$.
2. $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ iff $\mathrm{Dom}(\gamma_1) = \mathrm{Dom}(\gamma_2) = \mathrm{Dom}(\Gamma)$ and for every $u \in \mathrm{Dom}(\Gamma)$, $\Gamma'; \epsilon \vdash \gamma_1(u) = \gamma_2(u) : \gamma_1(\Gamma(u))$.

Using the typing rule for variables, we get $\Gamma \vdash \mathrm{id}_\Gamma : \Gamma$ for any context Γ. We can also prove the following lemma, which states that application of a substitution commutes with a single-point substitution on a variable not in its domain.

Lemma 1 (Independent Substitutions Commute)
If $\mathrm{Dom}(\sigma) \subseteq S$ *and* $u \notin S$ *and* $FV(M) \subseteq S$, *and* $FV(\mathcal{O}) \subseteq S \cup \{u\}$ *(where* $\mathcal{O}$ *is a kind, family or object term), then:*

$$(\sigma\mathcal{O})[\sigma M/u] = \sigma(\mathcal{O}[M/u])$$

Proof: Structural induction on $\mathcal{O}$.

■

3 Elementary Properties

Lemma 2 (Weakening of Intuitionistic Contexts) *Let J be any judgment.*

1. *If* $\Gamma, \Gamma' \vdash J$ *and* $u \notin \mathrm{Dom}(\Gamma)$ *then* $\Gamma, u{:}A, \Gamma' \vdash J$.
2. *If* $\Gamma, \Gamma'; \Delta \vdash J$ *and* $u \notin \mathrm{Dom}(\Gamma, \Gamma')$ *then* $\Gamma, u{:}A, \Gamma'; \Delta \vdash J$.

Lemma 3 (Reflexivity)

1. *If* $\Gamma; \Delta \vdash M : A$ *then* $\Gamma; \Delta \vdash M = M : A$.
2. *If* $\Gamma \vdash A : K$ *then* $\Gamma \vdash A = A : K$.
3. *If* $\Gamma \vdash K : \mathsf{kind}$ *then* $\Gamma \vdash K = K : \mathsf{kind}$.

HP prove a simple substitution lemma, which allows a single well-typed term to be substituted for a variable in the derivation of any judgment, and a simple functionality lemma which allows a single term to be substituted into each side of an equality judgment. Since our proof of injectivity makes heavy use of simultaneous substitutions of the form defined in Section 2.3, we will need more general substitution and functionality lemmas. These may be proved using the single-point versions, or they may be proved directly by induction on derivations. We have chosen the latter approach, and prove the single-point substitution and functionality lemmas as special cases.

Lemma 4 (Free Variables)

1. *If* $\Gamma \vdash J$, *then* $FV(J) \subseteq \mathrm{Dom}(\Gamma)$.
2. *If* $\Gamma; \Delta \vdash J$, *then* $FV(J) \subseteq \mathrm{Dom}(\Gamma) \cup \mathrm{Dom}(\Delta)$.

Proof: Trivial induction over derivations.

■

Lemma 5 (Extending Substitutions)

1. *If* $\Gamma' \vdash \gamma : \Gamma$ *and* $u \notin \mathrm{Dom}(\Gamma) \cup \mathrm{Dom}(\Gamma')$, *then* $\Gamma', u{:}\gamma A \vdash \gamma[u \mapsto u] : \Gamma, u{:}A$.
2. *If* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ *and* $u \notin \mathrm{Dom}(\Gamma) \cup \mathrm{Dom}(\Gamma')$, *then* $\Gamma', u{:}\gamma_1 A \vdash \gamma_1[u \mapsto u] = \gamma_2[u \mapsto u] : \Gamma, u{:}A$.

Proof: Direct, using the definition of well-formed and equal subsitutions. The proof requires Weakening (Lemma 2).

■

Lemma 6 (Intuitionistic Substitution)

1. *If* $\Gamma \vdash J$ *and* $\Gamma' \vdash \gamma : \Gamma$, *then* $\Gamma' \vdash \gamma J$.
2. *If* $\Gamma; \Delta \vdash J$ *and* $\mathrm{Dom}(\Gamma) \cap \mathrm{Dom}(\Delta) = \emptyset$ *and* $\Gamma' \vdash \gamma : \Gamma$, *then* $\Gamma'; \gamma\Delta \vdash \gamma J$.

Proof: By induction on derivations, using the preceding lemmas.

■

Corollary 1 (One-Point Intuitionistic Substitution) *Suppose* $\vdash \Gamma, u{:}A, \Gamma'\ \mathsf{context}$ *and* $\Gamma; \epsilon \vdash M : A$.

1. *If* $\Gamma, u{:}A, \Gamma' \vdash J$ *then* $\Gamma, \Gamma'[M/u] \vdash J[M/u]$.
2. *If* $\Gamma, u{:}A, \Gamma'; \Delta \vdash J$ *and* $\mathrm{Dom}(\Gamma, u{:}A, \Gamma') \cap \mathrm{Dom}(\Delta) = \emptyset$, *then* $\Gamma, \Gamma'[M/u]; \Delta[M/u] \vdash J[M/u]$.

Lemma 7 (Intuitionistic Context Conversion) *Assume* $\vdash \Gamma, u{:}A\ \mathsf{context}$ *and* $\Gamma \vdash B : \mathsf{Type}$ *and* $\Gamma \vdash A = B : \mathsf{Type}$.

1. *If* $\Gamma, u{:}A \vdash J$ *then* $\Gamma, u{:}B \vdash J$.
2. *If* $\Gamma, u{:}A; \Delta \vdash J$ *and* $\mathrm{Dom}(\Delta) \cap (\mathrm{Dom}(\Gamma) \cup \{u\}) = \emptyset$, *then* $\Gamma, u{:}B; \Delta \vdash J$.

Proof: Direct, using the preceding lemmas. The two parts are similar; we will only show part (2).
By rule, $\Gamma, u{:}B; \epsilon \vdash u : B$.
By symmetry and type conversion, $\Gamma, u{:}B; \epsilon \vdash u : A$.
By renaming, $\Gamma, v{:}A; \Delta[v/u] \vdash J[v/u]$.
By weakening (Lemma 2), $\Gamma, u{:}B, v{:}A; \Delta[v/u] \vdash J[v/u]$.
By substitution (Lemma 1), $\Gamma, u{:}B; \Delta[v/u][u/v] \vdash J[v/u][u/v]$.
That is, $\Gamma, u{:}B; \Delta \vdash J$.

■

Lemma 8 (Functionality for Typing) *Assume* $\Gamma' \vdash \gamma_1 : \Gamma$ *and* $\Gamma' \vdash \gamma_2 : \Gamma$ *and* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$.

1. *If* $\Gamma; \Delta \vdash P : B$ *and* $\mathrm{Dom}(\Gamma) \cap \mathrm{Dom}(\Delta) = \emptyset$, *then* $\Gamma'; \gamma_1\Delta \vdash \gamma_1 P = \gamma_2 P : \gamma_1 B$.
2. *If* $\Gamma \vdash B : K$ *then* $\Gamma' \vdash \gamma_1 B = \gamma_2 B : \gamma_1 K$.
3. *If* $\Gamma \vdash K : \mathsf{kind}$ *then* $\Gamma' \vdash \gamma_1 K = \gamma_2 K : \mathsf{kind}$.

Proof: By induction on derivations.

■

Corollary 2 (One-Point Functionality for Typing) *Assume* $\vdash \Gamma, u{:}A, \Gamma'$ $\mathsf{context}$, $\Gamma; \epsilon \vdash M = N : A$, $\Gamma; \epsilon \vdash M : A$ *and* $\Gamma; \epsilon \vdash N : A$.

1. *If* $\Gamma, u{:}A, \Gamma'; \Delta \vdash P : B$ *and* $\mathrm{Dom}(\Delta) \cap (\mathrm{Dom}(\Gamma, u{:}A, \Gamma') = \emptyset$ *then* $\Gamma, \Gamma'[M/u]; \Delta[M/u] \vdash P[M/u] = P[N/u] : B[M/u]$

2. *If* $\Gamma, u{:}A, \Gamma' \vdash B : K$ *then* $\Gamma, \Gamma'[M/u] \vdash B[M/u] = B[N/u] : K[M/u]$.

3. *If* $\Gamma, u{:}A, \Gamma' \vdash K : \mathsf{kind}$ *then* $\Gamma, \Gamma'[M/u] \vdash K[M/u] = K[N/u] : \mathsf{kind}$

Proof: Direct, from the preceding lemma.

■

Lemma 9 (Inversion on Simple Types and Kinds)

1. *If* $\Gamma \vdash \Pi u{:}A_1.A_2 : K$ *then* $\Gamma \vdash A_1 : \mathsf{Type}$ *and* $\Gamma, u{:}A_1 \vdash A_2 : \mathsf{Type}$.

2. *If* $\Gamma \vdash A \multimap B : K$ *then* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma \vdash B : \mathsf{Type}$.

3. *If* $\Gamma \vdash A \mathbin{\&} B : K$ *then* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma \vdash B : \mathsf{Type}$.

4. *If* $\Gamma \vdash \Pi u{:}A.K : \mathsf{kind}$ *then* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma, u{:}A \vdash K : \mathsf{kind}$.

Proof: Parts (1) through (3) are proved by induction on typing derivations. Part (4) is immediate by inspection of the kind formation rules.

■

We are now able to prove the important regularity property. This lemma states that all the terms that appear on the right-hand side of any judgment are well-formed, provided the context is valid—in other words, validity of what appears on the left of the turnstile implies validity of what appears on the right.

Lemma 10 (Regularity) *Assume* $\vdash \Gamma$ $\mathsf{context}$ *and* $\Gamma \vdash \Delta$ $\mathsf{context}$.

1. *If* $\Gamma; \Delta \vdash M : A$ *then* $\Gamma \vdash A : \mathsf{Type}$.

2. *If* $\Gamma; \Delta \vdash M = N : A$ *then* $\Gamma; \Delta \vdash M : A$ *and* $\Gamma; \Delta \vdash N : A$ *and* $\Gamma \vdash A : \mathsf{Type}$.

3. *If* $\Gamma \vdash A : K$ *then* $\Gamma \vdash K : \mathsf{kind}$.

4. *If* $\Gamma \vdash A = B : K$ *then* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : K$ *and* $\Gamma \vdash K : \mathsf{kind}$.

5. *If* $\Gamma \vdash K = L : \mathsf{kind}$ *then* $\Gamma \vdash K : \mathsf{kind}$ *and* $\Gamma \vdash L : \mathsf{kind}$.

Proof: By induction on derivations, using Corollary 2 and Lemma 9. The inversion lemma is required for elimination rules (and congruence rules for elimination forms) in order to establish well-formedness of the classifying type or kind. The functionality property is required for the cases involving applications of dependently-typed functions.

Case:

$$\frac{\Gamma; \Delta \vdash M_1 = M_2 : \Pi u{:}A.B \quad \Gamma; \epsilon \vdash N_1 = N_2 : A}{\Gamma; \Delta \vdash M_1\, N_1 = M_2\, N_2 : B[N_1/u]}$$

By the i.h. on the first subderivation, $\Gamma; \Delta \vdash M_1 : \Pi u{:}A.B$ and $\Gamma; \Delta \vdash M_2 : \Pi u{:}A.B$ and $\Gamma \vdash \Pi u{:}A.B : \mathsf{Type}$.
By Lemma 9, $\Gamma \vdash A : \mathsf{Type}$ and $\Gamma, u{:}A \vdash B : \mathsf{Type}$.
By the i.h. on the second subderivation, $\Gamma; \epsilon \vdash N_1 : A$ and $\Gamma; \epsilon \vdash N_2 : A$.
By Lemma 1, $\Gamma \vdash B[N_1/u] : \mathsf{Type}$, as required.
By rule, $\Gamma; \Delta \vdash M_1\, N_1 : B[N_1/u]$, as required.
By rule, $\Gamma; \Delta \vdash M_2\, N_2 : B[N_2/u]$.
By Corollary 2, $\Gamma \vdash B[N_1/u] = B[N_2/u] : \mathsf{Type}$.
By the symmetry and type conversion rules, $\Gamma; \Delta \vdash M_2\, N_2 : B[N_1/u]$, as required.

Case:
$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma, u{:}A; \Delta \vdash M_1 = M_2 : B}{\Gamma; \Delta \vdash \lambda u{:}A_1.M_1 = \lambda u{:}A_2.M_2 : \Pi u{:}A.B}$$

By the i.h. on the first two subderivations, $\Gamma \vdash A : \mathsf{Type}$ and $\Gamma \vdash A_1 : \mathsf{Type}$ and $\Gamma \vdash A_2 : \mathsf{Type}$.
By the i.h. on the third subderivation, $\Gamma, u{:}A; \Delta \vdash M_1 : B$ and $\Gamma, u{:}A; \Delta \vdash M_2 : B$ and $\Gamma, u{:}A \vdash B : \mathsf{Type}$.
By rule, $\Gamma \vdash \Pi u{:}A.B : \mathsf{Type}$, as required.
Using the symmetry rule and the context conversion lemma, $\Gamma, u{:}A_1; \Delta \vdash M_1 : B$ and $\Gamma, u{:}A_2; \Delta \vdash M_2 : B$.
By rule, $\Gamma; \Delta \vdash \lambda u{:}A_1.M_1 : \Pi u{:}A_1.B$ and $\Gamma; \Delta \vdash \lambda u{:}A_2.M_2 : \Pi u{:}A_2.B$.
By reflexivity, $\Gamma, u{:}A \vdash B = B : \mathsf{Type}$.
By rule, $\Gamma \vdash \Pi u{:}A.B = \Pi u{:}A_1.B : \mathsf{Type}$ and $\Gamma \vdash \Pi u{:}A.B = \Pi u{:}A_2.B : \mathsf{Type}$.
By symmetry and type conversion, $\Gamma; \Delta \vdash \lambda u{:}A_1.M_1 : \Pi u{:}A.B$ and $\Gamma; \Delta \vdash \lambda u{:}A_2.M_2 : \Pi u{:}A.B$, as required.
■

It is a simple matter to lift the regularity property for the object equality judgment to substitutions.

Corollary 3 (Regularity for Substitution Equality) *If* $\vdash \Gamma'$ context *and* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$*, then* $\Gamma' \vdash \gamma_1 : \Gamma$ *and* $\Gamma' \vdash \gamma_2 : \Gamma$.

Proof: Direct, using regularity and the definitions of well-formed and equal substitutions.
■

With regularity established, we can prove that performing equal substitutions on equal terms yields equal terms. This property is called Functionality, and we will use it often.

Lemma 11 (Functionality for Equality) *Assume* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$.

1. *If* $\Gamma; \Delta \vdash M = N : A$ *then* $\Gamma'; \gamma_1\Delta \vdash \gamma_1 M = \gamma_2 N : \gamma_1 A$.
2. *If* $\Gamma \vdash A = B : K$ *then* $\Gamma' \vdash \gamma_1 A = \gamma_2 B : \gamma_1 K$.
3. *If* $\Gamma \vdash K = L : \mathsf{kind}$ *then* $\Gamma' \vdash \gamma_1 K = \gamma_2 L : \mathsf{kind}$.

Proof: Direct. The three parts are similar, so we will only show the first one.
By Corollary 3, $\Gamma' \vdash \gamma_1 : \Gamma$.
By Lemma 6, $\Gamma', \gamma_1\Delta \vdash \gamma_1 M = \gamma_1 N : \gamma_1 A$.
By Lemma 10, $\Gamma; \Delta \vdash N : A$.
By Lemma 8, $\Gamma'; \gamma_1\Delta \vdash \gamma_1 N = \gamma_2 N : \gamma_1 A$.
By the transitivity rule, $\Gamma'; \gamma_1\Delta \vdash \gamma_1 M = \gamma_2 N : \gamma_1 A$, as required.
■

HP prove a one-point version of the functionality lemma, which is useful for us as well. It may either be proved as a special case of the previous lemma, or be proved directly using the one-point versions of the lemmas used in the proof just given.

Lemma 12 (One-Point Functionality for Equality) *Assume* $\vdash \Gamma, u{:}A, \Gamma'$ context *and* $\Gamma, u{:}A, \Gamma' \vdash \Delta$ context *and* $\Gamma; \epsilon \vdash M = N : A$.

1. *If* $\Gamma, u{:}A, \Gamma'; \Delta \vdash O = P : B$ *then* $\Gamma, \Gamma'[M/u]; \Delta[M/u] \vdash O[M/u] = P[N/u] : B[M/u]$.
2. *If* $\Gamma, u{:}A, \Gamma' \vdash B = C : K$ *then* $\Gamma, \Gamma'[M/u] \vdash B[M/u] = C[N/u] : K[M/u]$.
3. *If* $\Gamma, u{:}A, \Gamma' \vdash K = L : \mathsf{kind}$*, then* $\Gamma, \Gamma'[M/u] \vdash K[M/u] = L[N/u] : \mathsf{kind}$.

We can also use regularity to prove several inversion properties on typing derivations.

Lemma 13 (Typing Inversion) *Assume* $\vdash \Gamma$ context *and* $\Gamma \vdash \Delta$ context.

1. *If* $\Gamma; \Delta \vdash u : A$ *then either* $\Delta = \epsilon$ *and* $\Gamma(u) = B$ *and* $\Gamma \vdash A = B : \mathsf{Type}$ *or* $\Delta = (u\hat{:}B)$ *and* $\Gamma \vdash A = B : \mathsf{Type}$.
2. *If* $\Gamma; \Delta \vdash c : A$ *then* $\Delta = \epsilon$ *and* $S(c) = A'$ *and* $\Gamma \vdash A = A' : \mathsf{Type}$.

3. *If* $\Gamma; \Delta \vdash M\, N : A$ *then* $\Gamma; \Delta \vdash M : \Pi u{:}A_2.A_1$ *and* $\Gamma; \epsilon \vdash N : A_2$ *and* $\Gamma \vdash A_1[N/u] = A : \mathsf{Type}$.

4. *If* $\Gamma; \Delta \vdash \lambda u{:}A.M : B$, *then* $\Gamma \vdash B = \Pi u{:}A.A' : \mathsf{Type}$, $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma, u{:}A; \Delta \vdash M : A'$.

5. *If* $\Gamma; \Delta \vdash M\,\hat{}\,N : A$ *then* $\Delta = (\Delta_1, \Delta_2)$ *and* $\Gamma; \Delta_1 \vdash M : A_2 \multimap A_1$ *and* $\Gamma; \Delta_2 \vdash N : A_2$ *and* $\Gamma \vdash A_1 = A : \mathsf{Type}$.

6. *If* $\Gamma; \Delta \vdash \hat{\lambda} u{:}A.M : B$, *then* $\Gamma \vdash B = A \multimap A' : \mathsf{Type}$ *and* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma; \Delta, u\hat{:}A \vdash M : A'$.

7. *If* $\Gamma; \Delta \vdash \pi_i M : A$ *then* $\Gamma; \Delta \vdash M : A_1 \& A_2$ *and* $\Gamma \vdash A_i = A : \mathsf{Type}$.

8. *If* $\Gamma; \Delta \vdash \langle M_1, M_2 \rangle : A$ *then* $\Gamma \vdash A = A_1 \& A_2 : \mathsf{Type}$ *and* $\Gamma; \Delta \vdash M_1 : A_1$ *and* $\Gamma; \Delta \vdash M_2 : A_2$.

9. *If* $\Gamma \vdash \Pi u{:}A.B : K$ *then* $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ *and* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma, u{:}A \vdash B : \mathsf{Type}$.

10. *If* $\Gamma \vdash A \multimap B : K$ *then* $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ *and* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma \vdash B : \mathsf{Type}$.

11. *If* $\Gamma \vdash A \& B : K$ *then* $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ *and* $\Gamma \vdash A : \mathsf{Type}$ *and* $\Gamma \vdash B : \mathsf{Type}$.

12. *If* $\Gamma \vdash \top : K$ *then* $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$.

13. *If* $\Gamma \vdash a : K$ *then* $S(a) = L$ *and* $\Gamma \vdash K = L : \mathsf{kind}$.

14. *If* $\Gamma \vdash A\, M : K$, *then* $\Gamma \vdash A : \Pi u{:}A_2.K_1$ *and* $\Gamma; \epsilon \vdash M : A_2$ *and* $\Gamma \vdash K_1[M/u] = K : \mathsf{kind}$.

15. *If* $\Gamma \vdash \lambda u{:}A_1.A_2 : K$ *then* $\Gamma \vdash K = \Pi u{:}A_1.K' : \mathsf{kind}$ *and* $\Gamma \vdash A_1 : \mathsf{Type}$ *and* $\Gamma, u{:}A_1 \vdash A_2 : K'$.

16. *If* $\Gamma \vdash \Pi u{:}A_1.K_2 : \mathsf{kind}$ *then* $\Gamma \vdash A_1 : \mathsf{Type}$ *and* $\Gamma, u{:}A_1 \vdash K_2 : \mathsf{kind}$.

Proof: Straightforward, by induction on typing derivations.

■

We will also state here an inversion lemma for kind equality judgments. HP's equality inversion included families as well as kinds, but that version of the lemma does not hold in the presence of family-level λ-abstraction. Fortunately, HP's only use of equality inversion was in proving their Injectivity of Products lemma; we are going to prove injectivity separately, so we have no need for equality inversion on families at this stage.

Lemma 14 (Equality Inversion for Kinds)

1. *If* $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ *or* $\Gamma \vdash \mathsf{Type} = K : \mathsf{kind}$ *then* $K = \mathsf{Type}$.

2. *If* $\Gamma \vdash \Pi u{:}A.K_1 = L : \mathsf{kind}$ *or* $\Gamma \vdash L = \Pi u{:}A.K_1 : \mathsf{kind}$ *then* $L = \Pi u{:}B.L_1$ *where* $\Gamma \vdash A = B : \mathsf{Type}$ *and* $\Gamma, u{:}A \vdash K_1 = L_1 : \mathsf{kind}$.

Proof: Straightforward by induction on derivations, since there are no parallel conversion or extensionality rules for kinds.

■

4 Injectivity

The goal of this section is to prove a theorem that generalizes the "Injectivity of Products" lemma used by HP in the proofs of subject reduction and soundness of algorithmic equality. For Linear LF we must extend the idea of that lemma to all of the simple type constructors and so we will call it simply "injectivity."

Injectivity states that if two families formed using the same connective are definitionally equal, then corresponding subterms of those families will be equal as well. For example, if $\Gamma \vdash A_1 \multimap B_1 = A_2 \multimap B_2 : \mathsf{Type}$ then, according to injectivity, $\Gamma \vdash A_1 = A_2 : \mathsf{Type}$ and $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$. (In other words, the mapping from pairs of families to families embodied by each connective is one-to-one up to definitional equality.) HP are able to prove injectivity by means of an inversion lemma which is in turn proved by induction on derivations, but this is not possible in the presence of family-level λ-abstraction. The difficult case to prove by induction is transitivity: even if A and B have the same primary connective, if they are judged equal by virtue of their both being equal to C, then C may not have that same primary connective, rendering the induction hypothesis useless.

Fortunately, the only way this can happen is if one of the parallel conversion rules is used in the derivations of $A = C$ and $C = B$. This suggests that the appropriate strengthening of the induction hypothesis should apply not only to families with the same primary connective, but also to families that can be *converted* to ones with the same primary connective. We can state such a hypothesis in the form of a logical relation: we will define a relation on families that implies the injectivity property at base kind and extend it to higher kinds in the usual way. It will then be possible to prove that definitionally equal terms are related under this relation, and we will use this fact to prove the injectivity theorem.

To show that equal terms are related, we must strengthen the induction hypothesis once again, proving instead their images under equal substitutions are related. To do this we will make use of several properties of substitutions, and so before proving the main theorem of the section we will spend some time establishing our terminology for substitutions and the properties of them that we need.

4.1 A Logical Relation

In defining this logical relation, and later the equality algorithm, we will use the following definition of the *weak head reduction* relation $\xrightarrow{\text{wh}}$:

$$\frac{}{(\lambda u{:}A.M_1)\, M_2 \xrightarrow{\text{wh}} M_1[M_2/u]} \qquad \frac{}{(\hat{\lambda} u{:}A.M_1)\,\hat{}\, M_2 \xrightarrow{\text{wh}} M_1[M_2/u]} \qquad \frac{}{\pi_i\langle M_1, M_2\rangle \xrightarrow{\text{wh}} M_i}$$

$$\frac{M_1 \xrightarrow{\text{wh}} M_1'}{M_1\, M_2 \xrightarrow{\text{wh}} M_1'\, M_2} \qquad \frac{M_1 \xrightarrow{\text{wh}} M_1'}{M_1\,\hat{}\, M_2 \xrightarrow{\text{wh}} M_1'\,\hat{}\, M_2} \qquad \frac{M \xrightarrow{\text{wh}} M'}{\pi_i M \xrightarrow{\text{wh}} \pi_i M'}$$

$$\frac{}{(\lambda u{:}A_1.A_2)\, M \xrightarrow{\text{wh}} A_2[M/u]} \qquad \frac{A \xrightarrow{\text{wh}} A'}{A\, M \xrightarrow{\text{wh}} A'\, M}$$

The reflexive, transitive closure of $\xrightarrow{\text{wh}}$ will be denoted $\xrightarrow{\text{wh}}{}^*$. It will be important that weak head reduction is deterministic.

Lemma 15 (Determinacy of Weak Head Reduction)

1. *If $M \xrightarrow{\text{wh}} M'$ and $M \xrightarrow{\text{wh}} M''$ then $M' = M''$.*
2. *If $A \xrightarrow{\text{wh}} A'$ and $A \xrightarrow{\text{wh}} A''$ then $A' = A''$.*

Proof: By induction on weak head reduction derivations.

■

Now we can define the logical relation we will use to prove injectivity. Intuitively, we want the relation at base kind to imply the injectivity property we are aiming for. We strengthen this definition and require that the injectivity hold for all possible weak head contracta of the two families in question. Thus we will effectively prove that any family that is definitionally equal to, say, a $\multimap$-family will weak head reduce to one with definitionally equal domain and codomain. This rescues the problematic transitivity case by removing the sensitivity to the exact structure of the terms being compared. Our logical relation is defined as follows:

1. $\Gamma \vdash A_1 \approx A_2 : \mathsf{Type}$ iff all of the following hold:
    - $A_1 \xrightarrow{\text{wh}}{}^* \Pi u{:}B_1.C_1$ iff $A_2 \xrightarrow{\text{wh}}{}^* \Pi u{:}B_2 C_2$.
    - $A_1 \xrightarrow{\text{wh}}{}^* B_1 \multimap C_1$ iff $A_2 \xrightarrow{\text{wh}}{}^* B_2 \multimap C_2$.
    - $A_1 \xrightarrow{\text{wh}}{}^* B_1 \mathbin{\&} C_1$ iff $A_2 \xrightarrow{\text{wh}}{}^* B_2 \mathbin{\&} C_2$.

- If $A_1 \xrightarrow{\text{wh}}{}^* \Pi x{:}B_1.C_1$ and $A_2 \xrightarrow{\text{wh}}{}^* \Pi x{:}B_2 C_2$, then $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$ and $\Gamma, u{:}B_1 \vdash C_1 = C_2 : \mathsf{Type}$.
- If $A_1 \xrightarrow{\text{wh}}{}^* B_1 \multimap C_1$ and $A_2 \xrightarrow{\text{wh}}{}^* B_2 \multimap C_2$ then $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$ and $\Gamma \vdash C_1 = C_2 : \mathsf{Type}$.
- If $A_1 \xrightarrow{\text{wh}}{}^* B_1 \,\&\, C_1$ and $A_2 \xrightarrow{\text{wh}}{}^* B_2 \,\&\, C_2$ then $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$ and $\Gamma \vdash C_1 = C_2 : \mathsf{Type}$.

2. $\Gamma \vdash A_1 \approx A_2 : \Pi u{:}B.K$ iff $\forall M_1, M_2$, if $\Gamma; \epsilon \vdash M_1 = M_2 : B$ then $\Gamma \vdash A_1\, M_1 \approx A_2\, M_2 : K[M_1/u]$.

3. $\Gamma \vdash K_1 \approx K_2 : \mathsf{kind}$ iff $\forall A_1, A_2$, $\Gamma \vdash A_1 \approx A_2 : K_1$ iff $\Gamma \vdash A_1 \approx A_2 : K_2$.

The construction of this logical relation is unusual in that in the function case, the arguments are required to be definitionally equal, rather than logically equivalent. Also, there is no need to extend the context in the function case as in the Kripke logical relation we will use later to prove the completeness of our algorithm. This turns out to be sufficient (because the terms on which a family depends cannot affect the shape of its weak head normal form), and simplifies the proof considerably.

Note also that the definition does not require that families be definitionally equal to be related at base kind. That's not necessary, and might make closure under head expansion tricky to prove.

4.2 Definitionally Equal Terms are Logically Related

Several lemmas are required in order to show that all definitionally equal terms are related by the logical relation we have defined. For the most part, the proofs of these lemmas are independent of one another, and each one is relevant to a few particular cases in the main structural induction proof at the end of this subsection.

The first lemma will allow us to conclude that any family-level constant is logically related to itself.

Lemma 16 (Logically Related Paths) *If* $(a : \Pi u_1{:}A_1. \cdots .\Pi u_k{:}A_k.K) \in S$, *and* $\Gamma; \epsilon \vdash M_i = M_i' : A_i[M_1, \ldots, M_{i-1}/u_1, \ldots, u_{i-1}]$ *for each* i, *then* $\Gamma \vdash a\, M_1 \cdots M_k \approx a\, M_1' \cdots M_k' : K[M_1, \ldots, M_k/u_1, \ldots, u_k]$.

Proof: By induction on the number of Π's in K.

> **Case:** $K = \mathsf{Type}$. Since $(a\, M_1 \cdots M_k)$ and $(a\, M_1' \cdots M_k')$ are weak head normal forms, we have $\Gamma \vdash a\, M_1 \cdots M_k \approx a\, M_1' \cdots M_k' : \mathsf{Type}$.
>
> **Case:** $K = \Pi v{:}B.K'$. Suppose that $\Gamma; \epsilon \vdash N = N' : B[M_1, \ldots, M_k/u_1, \ldots, u_k]$. We need to show that $\Gamma \vdash a\, M_1 \cdots M_k\, N \approx a\, M_1' \cdots M_k'\, N' : K'[M_1, \ldots, M_k, N/u_1, \ldots, u_k, v]$. But K' has fewer Π's in it than K, so this follows by the induction hypothesis.

■

We need the logical relations to be closed under head expansion in order to show that families proven equal by the parallel conversion rule are logically related. This fact is also used in the case for the congruence rule on λ-abstractions.

Lemma 17 (Closure under Head Expansion) *If* $\Gamma \vdash A \approx B : K$ *and* $\Gamma \vdash A' : K$ *and* $\Gamma \vdash B' : K$ *and* $A' \xrightarrow{\text{wh}}{}^* A$ *and* $B' \xrightarrow{\text{wh}}{}^* B$, *then* $\Gamma \vdash A' \approx B' : K$.

Proof: By induction on the size (number of Π's) of K.

> **Case:** $K = \mathsf{Type}$.
> Suppose $A' \xrightarrow{\text{wh}}{}^* \Pi u{:}A_1.B_1$.
> Using Lemma 15, $A \xrightarrow{\text{wh}}{}^* \Pi u{:}A_1.B_1$.
> By definition of $\approx$, B reduces to a Π-family and therefore so does B'.
> So, suppose $A' \xrightarrow{\text{wh}}{}^* \Pi u{:}C_1.D_1$ and $B' \xrightarrow{\text{wh}}{}^* \Pi u{:}C_2 D_2$.
> Using Lemma 15, $A \xrightarrow{\text{wh}}{}^* \Pi u{:}C_1.D_1$ and $B \xrightarrow{\text{wh}}{}^* \Pi u{:}C_2 D_2$.
> By definition of $\approx$, $\Gamma \vdash C_1 = C_2 : \mathsf{Type}$ and $\Gamma, u{:}C_1 \vdash D_1 = D_2 : \mathsf{Type}$.
> Similarly, the parts of the definition of $\approx$ pertaining to $\multimap$ and $\&$ are satisfied.
> So, $\Gamma \vdash A' \approx B' : \mathsf{Type}$.

Case: $K = \Pi u{:}C.K'$.
Suppose $\Gamma; \epsilon \vdash M_1 = M_2 : C$.
By definition of $\approx$, $\Gamma \vdash A\,M_1 \approx B\,M_2 : K'[M_1/u]$.
Observe that $A'\,M_1 \xrightarrow{\text{wh}} A\,M_1$ and $B'\,M_2 \xrightarrow{\text{wh}} B\,M_2$ and $K'[M_1/u]$ is smaller than K.
Thus by the i.h., $\Gamma \vdash A'\,M_1 \approx B'\,M_2 : K'[M_1/u]$.
So, by definition of $\approx$, $\Gamma \vdash A' \approx B' : K$.

■

The next lemma, which allows equal substitutions to be extended to map a new variable to equal terms, will be required for some of the later lemmas leading up to the main lemma of this section, as well as for the main lemma itself.

Lemma 18 *If* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ *and* $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_1 A$ *and* $\vdash \Gamma$ $\mathsf{context}$ *and* $u \notin \mathrm{Dom}(\Gamma)$*, then* $\Gamma' \vdash \gamma_1[u{\mapsto}M_1] = \gamma_2[u{\mapsto}M_2] : \Gamma, u{:}A$.

The next thing we need to establish is that the logical relation on families is symmetric. (Symmetry of the relation on kinds is obvious.) To overcome the asymmetry in the way substitutions are applied to kinds in the definition of the relation, we first prove the following lemma which states that performing equal substitutions on a kind will produce results that are logically related.

Lemma 19 (Equivalent Substitutions of a Valid Kind are Logically Related) *If* $\Gamma \vdash K : \mathsf{kind}$ *and* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ *and* $\vdash \Gamma$ $\mathsf{context}$ *and* $\Gamma' \vdash A \approx B : \gamma_1 K$*, then* $\Gamma' \vdash A \approx B : \gamma_2 K$.

Proof: By induction on the derivation of $\Gamma \vdash K : \mathsf{kind}$.

Case: $\Gamma \vdash K : \mathsf{kind}$ because $K = \mathsf{Type}$. Then $\gamma_1 K = \gamma_2 K$, so the lemma trivially holds.

Case: $\Gamma \vdash K : \mathsf{kind}$ because $K = \Pi x{:}C.K'$, $\Gamma \vdash C : \mathsf{Type}$ and $\Gamma, x{:}C \vdash K' : \mathsf{kind}$.

WLOG, we may assume $x \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Gamma')$, so $\vdash \Gamma, x{:}C$ $\mathsf{context}$.
Suppose that $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_2 C$. We need to show that $\Gamma' \vdash A\,M_1 \approx B\,M_2 : (\gamma_2 K')[M_1/x]$.
By Lemma 8, $\Gamma' \vdash \gamma_2 C = \gamma_1 C : \mathsf{Type}$, since equality of substitutions is symmetric.
By the type conversion rule, $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_1 C$.
From the assumption, $\Gamma' \vdash A\,M_1 \approx B\,M_2 : (\gamma_1 K')[M_1/x]$.
For $i = 1, 2$, let $\gamma_i' = \gamma_i[x{\mapsto}M_1]$.
Now, $\Gamma' \vdash A\,M_1 \approx B\,M_2 : \gamma_1' K'$.
By symmetry and transitivity, $\Gamma'; \epsilon \vdash M_1 = M_1 : \gamma_1 C$.
By Lemma 18, $\Gamma' \vdash \gamma_1' = \gamma_2' : \Gamma, x{:}C$.
By the induction hypothesis, $\Gamma' \vdash A\,M_1 \approx B\,M_2 : \gamma_2' K'$.
That is, $\Gamma' \vdash A\,M_1 \approx B\,M_2 : (\gamma_2 K')[M_1/x]$, which is what we needed.

■

Now we can prove that the logical relation is symmetric.

Lemma 20 (Symmetry of the Logical Relations) *If* $\Gamma \vdash K : \mathsf{kind}$ *and* $\vdash \Gamma$ $\mathsf{context}$ *and* $\Gamma \vdash B \approx A : K$*, then* $\Gamma \vdash A \approx B : K$.

Proof: By induction on the size (number of Π's) of K.

Case: $K = \mathsf{Type}$. There are several things to show. We will show the parts concerning reduction to Π-families; the others are similar.

- A weak head reduces to a Π-family (or a $\multimap$-family or a &-family) iff B does, by definition of $\approx$ and the assumption that $\Gamma \vdash B \approx A : \mathsf{Type}$.

- Suppose that $A \xrightarrow{\text{wh}}{}^{*} \Pi u{:}C_1.D_1$ and $B \xrightarrow{\text{wh}}{}^{*} \Pi u{:}C_2.D_2$. WLOG, we may assume $u \notin \text{Dom}(\Gamma)$.
  By the assumption, $\Gamma \vdash C_2 = C_1 : \mathsf{Type}$.
  By the symmetry rule, $\Gamma \vdash C_1 = C_2 : \mathsf{Type}$, as required.
  By the assumption, $\Gamma, u{:}C_2 \vdash D_2 = D_1 : \mathsf{Type}$.
  By Lemma 10, $\Gamma \vdash C_1 : \mathsf{Type}$ and $\Gamma \vdash C_2 : \mathsf{Type}$.
  Hence, $\vdash \Gamma, u{:}C_2$ $\mathsf{context}$.
  By Lemma 7, $\Gamma, u{:}C_1 \vdash D_2 = D_1 : \mathsf{Type}$.
  by the symmetry rule, $\Gamma, u{:}C_1 \vdash D_1 = D_2 : \mathsf{Type}$, as required.

Case: $K = \Pi u{:}C.K'$.
Suppose $\Gamma; \epsilon \vdash M_1 = M_2 : C$.
By the symmetry rule, $\Gamma; \epsilon \vdash M_2 = M_1 : C$.
By definition of $\approx$, $\Gamma \vdash B\,M_2 \approx A\,M_1 : K'[M_2/u]$.
By the i.h., $\Gamma \vdash A\,M_1 \approx B\,M_2 : K'[M_2/u]$.
Using Lemma 3, $\Gamma \vdash \text{id}_\Gamma = \text{id}_\Gamma : \Gamma$.
By Lemma 18, $\Gamma \vdash \text{id}_\Gamma[u \mapsto M_2] = \text{id}_\Gamma[u \mapsto M_1] : \Gamma, u{:}C$.
By Lemma 13, $\Gamma \vdash C : \mathsf{Type}$ and $\Gamma, u{:}C \vdash K' : \mathsf{kind}$.
Hence, $\vdash \Gamma, u{:}C$ $\mathsf{context}$.
By Lemma 19, $\Gamma \vdash A\,M_1 \approx B\,M_2 : K'[M_1/u]$.
Hence by definition of $\approx$, $\Gamma \vdash A \approx B : K$.

■

Finally, the following lemma handles the transitivity case in the proof of the main theorem of this section.

Lemma 21 (Transitivity of the Logical Relations) *If* $\vdash \Gamma$ $\mathsf{context}$ *and* $\Gamma \vdash A \approx C : K$ *and* $\Gamma \vdash C \approx B : K$*, then* $\Gamma \vdash A \approx B : K$.

Proof: By induction on the size (number of Π's) of K.

Case: $K = \mathsf{Type}$. There are several things to show. The parts concerning reduction to $\multimap$- and &-families are similar to those for Π-families, so we omit them here.

- Suppose B weak head reduces to a Π-family. By assumption, so does C. By the other assumption, so does A.
  Similarly, if A weak head reduces to a Π-family, so does B.
- Suppose that $A \xrightarrow{\text{wh}}{}^{*} \Pi u{:}D_1 E_1$ and $B \xrightarrow{\text{wh}}{}^{*} \Pi u{:}D_2 E_2$.
  (WLOG, we may assume $u \notin \text{Dom}(\Gamma)$.)
  By an assumption, $C \xrightarrow{\text{wh}}{}^{*} \Pi u{:}D_3 E_3$.
  Also, $\Gamma \vdash D_1 = D_3 : \mathsf{Type}$ and $\Gamma \vdash D_3 = D_2 : \mathsf{Type}$.
  By the transitivity rule, $\Gamma \vdash D_1 = D_2 : \mathsf{Type}$, as required.
  By the assumptions again, $\Gamma, u{:}D_1 \vdash E_1 = E_3 : \mathsf{Type}$ and $\Gamma, u{:}D_3 \vdash E_3 = E_2 : \mathsf{Type}$.
  By Lemma 10, $\Gamma \vdash D_1 : \mathsf{Type}$ and $\Gamma \vdash D_3 : \mathsf{Type}$.
  Hence, $\vdash \Gamma, u{:}D_3$ $\mathsf{context}$.
  By Lemma 7, $\Gamma, u{:}D_1 \vdash E_3 = E_2 : \mathsf{Type}$.
  By the transitivity rule, $\Gamma, u{:}D_1 \vdash E_1 = E_2 : \mathsf{Type}$.

Case: $K = \Pi u{:}F.K'$.
Suppose $\Gamma; \epsilon \vdash M_1 = M_2 : F$.
By Lemma 10, $\Gamma; \epsilon \vdash M_1 : F$.
By the reflexivity rule, $\Gamma; \epsilon \vdash M_1 = M_1 : F$.
By the assumptions, $\Gamma \vdash A\,M_1 \approx C\,M_1 : K'[M_1/u]$ and $\Gamma \vdash C\,M_1 \approx B\,M_2 : K'[M_1/u]$.
By the i.h., $\Gamma \vdash A\,M_1 \approx B\,M_2 : K'[M_1/u]$, as required.

■

Now we can prove that all definitionally equal families and kinds are logically related under equal substitutions. Since identity substitutions are equal to themselves, this implies that definitionally equal families and kinds are logically related. The injectivity property follows easily.

Lemma 22 (Definitionally Equal Terms are Logically Related under Subsitutions)

1. *If* $\Gamma \vdash A_1 = A_2 : K$ *and* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ *and* $\vdash \Gamma\ \mathsf{context}$ *then* $\Gamma' \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \gamma_1 K$.
2. *If* $\Gamma \vdash K_1 = K_2 : \mathsf{kind}$ *and* $\Gamma' \vdash \gamma_1 = \gamma_2 : \Gamma$ *and* $\vdash \Gamma\ \mathsf{context}$ *then* $\Gamma' \vdash \gamma_1 K_1 \approx \gamma_2 K_2 : \mathsf{kind}$.

Proof: By induction on derivations.

Case:

$$\frac{}{\Gamma \vdash a = a : K}\ (S(a) = K)$$

Since S is well-formed, K is closed.
By Lemma 16, $\Gamma' \vdash a \approx a : K$.
Since a and K are closed, $\Gamma' \vdash \gamma_1 a \approx \gamma_2 a : \gamma_1 K$.

Case:

$$\frac{}{\Gamma \vdash \top = \top : \mathsf{Type}}$$

Note that $\gamma_i \top = \top$ and $\gamma_i \mathsf{Type} = \mathsf{Type}$.
Also, $\top$ cannot weak-head reduce to anything but itself.
Thus by definition, $\Gamma \vdash \gamma_1 \top \approx \gamma_2 \top : \gamma_1 \mathsf{Type}$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash \Pi u{:}A_1.B_1 = \Pi u{:}A_2.B_2 : \mathsf{Type}}$$

WLOG, assume $u \notin \mathrm{Dom}(\Gamma')$.
Need to show $\Gamma' \vdash \Pi u{:}\gamma_1 A_1.\gamma_1 B_1 \approx \Pi u{:}\gamma_2 A_2.\gamma_2 B_2 : \mathsf{Type}$.
Observe that both of these trivially reduce to Π-families, and that neither will ever reduce to a $\multimap$- or &-family.
Also, neither may reduce to any Π-family but itself; that is, if $\Pi u{:}\gamma_i A_i.\gamma_i B_i \xrightarrow{\text{wh}}{}^{*} \Pi u{:}C_i.D_i$ then $C_i = \gamma_i A_i$ and $D_i = \gamma_i B_i$.
So, we must show that $\Gamma' \vdash \gamma_1 A_1 = \gamma_2 A_2 : \mathsf{Type}$ and $\Gamma, u{:}\gamma_1 A_1 \vdash \gamma_1 B_1 = \gamma_2 B_2 : \mathsf{Type}$.
By Lemma 11, $\Gamma' \vdash \gamma_1 A_1 = \gamma_2 A_2 : \mathsf{Type}$, as required.
By Lemma 10, $\Gamma' \vdash \gamma_1 A_1 : \mathsf{Type}$. Thus $\vdash \Gamma', u{:}\gamma_1 A_1\ \mathsf{context}$.
By Lemma 10, $\Gamma \vdash A_1 : \mathsf{Type}$.
Thus, $\vdash \Gamma, u{:}A\ \mathsf{context}$.
By Lemma 5, $\Gamma', u{:}\gamma_1 A_1 \vdash \gamma_1[u{\mapsto}u] = \gamma_2[u{\mapsto}u] : \Gamma, u{:}A_1$.
By Lemma 11, $\Gamma', u{:}\gamma_1 A_1 \vdash \gamma_1[u{\mapsto}u]B_1 = \gamma_2[u{\mapsto}u]B_2 : \mathsf{Type}$.
That is, $\Gamma', u{:}\gamma_1 A \vdash \gamma_1 B_1 = \gamma_2 B_2 : \mathsf{Type}$, as required.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \multimap B_1 = A_2 \multimap B_2 : \mathsf{Type}}$$

We need to show $\Gamma' \vdash \gamma_1 A_1 \multimap \gamma_1 B_1 \approx \gamma_2 A_2 \multimap \gamma_2 B_2 : \mathsf{Type}$.
Observe that both of these trivially reduce to a $\multimap$-family, and that neither will ever reduce to a Π- or &-family.
In fact, neither may reduce to any family but itself; thus, if $\gamma_i A_i \multimap \gamma_i B_i \xrightarrow{\text{wh}}{}^{*} C_i \multimap D_i$ then $C_i = \gamma_i A_i$ and $D_i = \gamma_i B_i$.
So, we must show that $\Gamma' \vdash \gamma_1 A_1 = \gamma_2 A_2 : \mathsf{Type}$ and $\Gamma' \vdash \gamma_1 B_1 = \gamma_2 B_2 : \mathsf{Type}$.
This follows by Lemma 11.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \,\&\, B_1 = A_2 \,\&\, B_2 : \mathsf{Type}}$$

Similar to the previous case.

Case:
$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma, u{:}A \vdash B_1 = B_2 : K}{\Gamma \vdash \lambda u{:}A_1.B_1 = \lambda u{:}A_2.B_2 : \Pi u{:}A.K}$$

WLOG, assume $u \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Gamma')$.
Need to show $\Gamma' \vdash \lambda u{:}\gamma_1 A_1.\gamma_1 B_1 \approx \lambda u{:}\gamma_2 A_2.\gamma_2 B_2 : \Pi u{:}\gamma_1 A.\gamma_1 K$.
So, suppose $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_1 A$.
Since $\Gamma \vdash A : \mathsf{Type}$, we know $\vdash \Gamma, u{:}A$ $\mathsf{context}$.
By Lemma 18, $\Gamma' \vdash \gamma_1[u \mapsto M_1] = \gamma_2[u \mapsto M_2] : \Gamma, u{:}A$.
By the i.h., $\Gamma' \vdash \gamma_1[u \mapsto M_1]B_1 \approx \gamma_2[u \mapsto M_2]B_2 : \gamma_1[u \mapsto M_1]K$.
That is, $\Gamma' \vdash (\gamma_1 B_1)[M_1/u] \approx (\gamma_2 B_2)[M_2/u] : \gamma_1 K[M_1/u]$.
Now, note that $(\lambda u{:}\gamma_i A_i.\gamma_i B_i)\, M_i \xrightarrow{\text{wh}} (\gamma_i B_i)[M_i/u]$.
Thus by Lemma 17, $\Gamma' \vdash (\lambda u{:}\gamma_1 A_1.\gamma_1 B_1)\, M_1 \approx (\lambda u{:}\gamma_2 A_2.\gamma_2 B_2)\, M_2 : \gamma_1 K[M_1/u]$.
By definition of $\approx$, $\Gamma' \vdash \lambda u{:}\gamma_1 A_1.\gamma_1 B_1 \approx \lambda u{:}\gamma_2 A_2.\gamma_2 B_2 : \Pi u{:}\gamma_1 A.\gamma_1 K$.

$$\frac{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K \quad \Gamma; \epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash A_1\, M_1 = A_2\, M_2 : K[M_1/u]}$$

By the i.h., $\Gamma' \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \Pi u{:}\gamma_1 B.\gamma_1 K$.
By Lemma 11, $\Gamma'; \epsilon \vdash \gamma_1 M_1 = \gamma_2 M_2 : \gamma_1 B$.
By definition of $\approx$, $\Gamma' \vdash (\gamma_1 A_1)\,(\gamma_1 M_1) \approx (\gamma_2 A_2)\,(\gamma_2 M_2) : \gamma_1 K[\gamma_1 M_1/u]$.
That is, $\Gamma' \vdash \gamma_1(A_1\, M_1) \approx \gamma_2(A_2\, M_2) : \gamma_1(K[M_1/u])$.

Case:
$$\frac{\Gamma \vdash B = A : K}{\Gamma \vdash A = B : K}$$

Since equality of substitutions is symmetric, $\Gamma' \vdash \gamma_2 = \gamma_1 : \Gamma$.
By the i.h., $\Gamma' \vdash \gamma_2 B \approx \gamma_1 A : \gamma_2 K$.
By Lemma 20, $\Gamma' \vdash \gamma_1 A \approx \gamma_2 B : \gamma_2 K$.
By Lemma 19, $\Gamma' \vdash \gamma_1 A \approx \gamma_2 B : \gamma_1 K$.

Case:
$$\frac{\Gamma \vdash A = C : K \quad \Gamma \vdash C = B : K}{\Gamma \vdash A = B : K}$$

Using symmetry and transitivity, $\Gamma' \vdash \gamma_1 = \gamma_1 : \Gamma$.
By the i.h., $\Gamma' \vdash \gamma_1 A \approx \gamma_1 C : \gamma_1 K$ and $\Gamma' \vdash \gamma_1 C \approx \gamma_2 B : \gamma_1 K$.
By Lemma 21, $\Gamma \vdash \gamma_1 A \approx \gamma_2 B : \gamma_1 K$.

Case:
$$\frac{\Gamma \vdash B : \mathsf{Type} \quad \Gamma, u{:}B \vdash A_1 = A_2 : K \quad \Gamma; \epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash (\lambda u{:}B.A_1)\, M_1 = A_2[M_2/u] : K[M_1/u]}$$

WLOG, assume $u \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Gamma')$.
Need to show $\Gamma' \vdash (\lambda u{:}\gamma_1 B.\gamma_1 A_1)\, \gamma_1 M_1 \approx (\gamma_2 A_2)[\gamma_2 M_2/u] : (\gamma_1 K)[\gamma_1 M_1/u]$.
By Lemma 11, $\Gamma'; \epsilon \vdash \gamma_1 M_1 = \gamma_2 M_2 : \gamma_1 B$.
Since $\Gamma \vdash B : \mathsf{Type}$, we have $\vdash \Gamma, x{:}B$ $\mathsf{context}$.
By A LEMMA, $\Gamma' \vdash \gamma_1[u \mapsto \gamma_1 M_1] = \gamma_2[u \mapsto \gamma_2 M_2] : \Gamma, u{:}B$.
By the i.h., $\Gamma' \vdash \gamma_1[u \mapsto \gamma_1 M_1]A_1 \approx \gamma_2[u \mapsto \gamma_2 M_2]A_2 : \gamma_1[u \mapsto \gamma_1 M_1]K$.
That is, $\Gamma' \vdash (\gamma_1 A_1)[\gamma_1 M_1/u] \approx (\gamma_2 A_2)[\gamma_2 M_2/u] : (\gamma_1 K)[\gamma_1 M_1/u]$.
Note that $(\lambda u{:}\gamma_1 B.\gamma_1 A_1)\, \gamma_1 M_1 \xrightarrow{\text{wh}} (\gamma_1 A_1)[\gamma_1 M_1/u]$.
Thus by Lemma 17, $\Gamma' \vdash (\lambda u{:}\gamma_1 B.\gamma_1 A_1)\, \gamma_1 M_1 \approx (\gamma_2 A_2)[\gamma_2 M_2/u] : (\gamma_1 K)[\gamma_1 M_1/u]$.

Case:

$$\frac{\Gamma \vdash B : \mathsf{Type} \quad \Gamma \vdash A_1 : \Pi u{:}B.K \quad \Gamma \vdash A_2 : \Pi u{:}B.K \quad \Gamma, u{:}B \vdash A_1\,u = A_2\,u : K}{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K}$$

WLOG, assume $u \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Gamma')$.
Need to show $\Gamma' \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \Pi u{:}\gamma_1 B.\gamma_1 K$.
So, suppose, $\Gamma' \vdash M_1 = M_2 : \gamma_1 B$.
Since $\Gamma \vdash B : \mathsf{Type}$, we have $\vdash \Gamma, u{:}B$ $\mathsf{context}$.
By Lemma 18, $\Gamma' \vdash \gamma_1[u{\mapsto}M_1] = \gamma_2[u{\mapsto}M_2] : \Gamma, u{:}B$.
By the i.h., $\Gamma' \vdash \gamma_1[u{\mapsto}M_1](A_1\,u) \approx \gamma_2[u{\mapsto}M_2](A_2\,u) : \gamma_1[u{\mapsto}M_1]K$.
That is, $\Gamma' \vdash (\gamma_1 A_1)\,M_1 \approx (\gamma_2 A_2)\,M_2 : \gamma_1 K$.
By definition of $\approx$, $\Gamma' \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \Pi u{:}\gamma_1 B.\gamma_1 K$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : K' \quad \Gamma \vdash K' = K : \mathsf{kind}}{\Gamma \vdash A_1 = A_2 : K}$$

Using symmetry and transitivity, $\Gamma' \vdash \gamma_1 = \gamma_1 : \Gamma$.
By the i.h., $\Gamma' \vdash \gamma_1 K' \approx \gamma_1 K : \mathsf{kind}$ and $\Gamma' \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \gamma_1 K'$.
By definition of $\approx$ for kinds, $\Gamma \vdash \gamma_1 A_1 \approx \gamma_2 A_2 : \gamma_1 K$.

Case:

$$\frac{}{\Gamma \vdash \mathsf{Type} = \mathsf{Type} : \mathsf{kind}}$$

Note that $\gamma_1\mathsf{Type} = \gamma_2\mathsf{Type} = \mathsf{Type}$.
Trivially, $\Gamma' \vdash A \approx B : \mathsf{Type}$ iff $\Gamma' \vdash A \approx B : \mathsf{Type}$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash K_1 = K_2 : \mathsf{kind}}{\Gamma \vdash \Pi u{:}A_1.K_1 = \Pi u{:}A.K_2 : \mathsf{kind}}$$

WLOG, assume $u \notin \mathrm{Dom}(\Gamma), \mathrm{Dom}(\Gamma')$.
Need to show $\Gamma' \vdash \Pi u{:}\gamma_1 A_1.\gamma_1 K_1 \approx \Pi u{:}\gamma_2 A_2.\gamma_2 K_2 : \mathsf{kind}$,
i.e., $\Gamma' \vdash C_1 \approx C_2 : \Pi u{:}\gamma_1 A_1.\gamma_1 K_1$ iff $\Gamma' \vdash C_1 \approx C_2 : \Pi u{:}\gamma_2 A_2.\gamma_2 K_2$.
(We will only show one direction. The other is similar, except for one additional use of the symmetry rule for family equality.)
Suppose $\Gamma' \vdash C_1 \approx C_2 : \Pi u{:}\gamma_2 A_2.\gamma_2 K_2$.
Need to show $\Gamma' \vdash C_1 \approx C_2 : \Pi u{:}\gamma_1 A_1.\gamma_1 K_1$.
So, suppose $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_1 A_1$.
Need to show $\Gamma' \vdash C_1\,M_1 \approx C_2\,M_2 : (\gamma_1 K_1)[M_1/u]$.
By Lemma 1, $\Gamma' \vdash \gamma_1 A_1 = \gamma_2 A_2 : \mathsf{Type}$.
By the type conversion rule, $\Gamma'; \epsilon \vdash M_1 = M_2 : \gamma_2 A_2$.
Using an assumption, $\Gamma' \vdash C_1\,M_1 \approx C_2\,M_2 : (\gamma_2 K_2)[M_1/u]$.
By Lemma 10, $\Gamma \vdash A_1 : \mathsf{Type}$, so $\vdash \Gamma, u{:}A_1$ $\mathsf{context}$.
For $i = 1, 2$, define $\gamma_i' = \gamma_i[u{\mapsto}M_1]$. Then $(\gamma_i K_i)[M_1/u] = \gamma_i' K_i$.
Using regularity, reflexivity, and Lemma 18, $\Gamma' \vdash \gamma_1' = \gamma_2' : \Gamma, u{:}A_1$.
By the I.H., $\Gamma' \vdash \gamma_1' K_1 \approx \gamma_2' K_2 : \mathsf{kind}$.
Therefore, since $\Gamma' \vdash C_1\,M_1 \approx C_2\,M_2 : \gamma_2' K_2$, $\Gamma' \vdash C_1\,M_1 \approx C_2\,M_2 : \gamma_1' K_1$.
That is, $\Gamma' \vdash C_1\,M_1 \approx C_2\,M_2 : (\gamma_1 K_1)[M_1/u]$ as required.

Case:

$$\frac{\Gamma \vdash K_2 = K_1 : \mathsf{kind}}{\Gamma \vdash K_1 = K_2 : \mathsf{kind}}$$

Using regularity, symmetry, substitution and type conversion, $\Gamma' \vdash \gamma_2 = \gamma_1 : \Gamma$.
By the i.h., $\Gamma \vdash \gamma_2 K_2 \approx \gamma_1 K_1 : \mathsf{kind}$.

So $\Gamma' \vdash A \approx B : \gamma_2 K_2$ iff $\Gamma' \vdash A \approx B : \gamma_1 K_1$.
In other words, $\Gamma' \vdash \gamma_1 K_1 \approx \gamma_2 K_2 : \mathsf{kind}$.

Case:

$$\frac{\Gamma \vdash K_1 = K_3 : \mathsf{kind} \quad \Gamma \vdash K_3 = K_2 : \mathsf{kind}}{\Gamma \vdash K_1 = K_2 : \mathsf{kind}}$$

Using various lemmas, $\Gamma' \vdash \gamma_1 = \gamma_1 : \Gamma$.
By the i.h., $\Gamma' \vdash \gamma_1 K_1 \approx \gamma_1 K_3 : \mathsf{kind}$ and $\Gamma' \vdash \gamma_1 K_3 \approx \gamma_2 K_2 : \mathsf{kind}$.
So, $\Gamma' \vdash A \approx B : \gamma_1 K_1$ iff $\Gamma' \vdash A \approx B : \gamma_1 K_3$ iff $\Gamma' \vdash A \approx B : \gamma_2 K_2$.
So, $\Gamma' \vdash \gamma_1 K_1 \approx \gamma_2 K_2 : \mathsf{kind}$.

■

Theorem 1 (Injectivity)

1. *If* $\Gamma \vdash \Pi u{:}A_1.A_2 = \Pi u{:}B_1.B_2 : \mathsf{Type}$ *and* $\vdash \Gamma\ \mathsf{context}$ *then* $\Gamma \vdash A_1 = B_1 : \mathsf{Type}$ *and* $\Gamma, u{:}A_1 \vdash A_2 = B_2 : \mathsf{Type}$.
2. *If* $\Gamma \vdash A_1 \multimap A_2 = B_1 \multimap B_2 : \mathsf{Type}$*and* $\vdash \Gamma\ \mathsf{context}$ *then* $\Gamma \vdash A_1 = B_1 : \mathsf{Type}$ *and* $\Gamma \vdash A_2 = B_2 : \mathsf{Type}$.
3. *If* $\Gamma \vdash A_1 \,\&\, A_2 = B_1 \,\&\, B_2 : \mathsf{Type}$ *and* $\vdash \Gamma\ \mathsf{context}$ *then* $\Gamma \vdash A_1 = B_1 : \mathsf{Type}$ *and* $\Gamma \vdash A_2 = B_2 : \mathsf{Type}$.

Proof: Direct, using Lemma 22 and the definition of $\approx$.

■

5 Equality Algorithm

5.1 Erasure

To avoid serious difficulties with dependencies on terms, the algorithm and Kripke logical relation presented by HP use *simple types* and *simple kinds* in place of ordinary families and kinds. Not only are the type-directed phase of HP's algorithm directed by simple types and the Kripke logical relation indexed by simple types and kinds, but the contexts (or "worlds") in both the algorithmic judgments and the logical relation give only simple types to variables. The process of erasing ordinary families and kinds into simple ones effectively identifies types that differ only in the terms that appear in them.

We adopt this practice of erasure as well, extending it to erase the distinction between intuitionistic and linear assumptions in a context. The need for this arises because of the splitting of the linear context that occurs in the typing and definitional equality rules for linear function applications. If this context-splitting were to be enforced in the algorithmic judgments, then the transitivity proof for those judgments would be upset by the possibility that different derivations mentioning the same linear application term might split the context differently.

Essentially, we want to avoid this problem by not requiring the algorithmic equality rule for linear applications to split the linear context. Such a change by itself would destroy the property that every linear assumption in a judgment must be used, so we also have to remove the restriction on linear variable use. However, this leaves us with two separate contexts that are treated in exactly the same way, so we go a step further and combine the intuitionistic and linear contexts into one. Consequently there is no distinction between intuitionistic and linear assumptions in the algorithm or logical relation. This does not affect soundness, since we only wish to prove the algorithm sound for well-typed terms, which must respect linearity.

5.1.1 Grammar for Simple Kinds, Types and Contexts

Our grammar for families and kinds with no term dependencies, and contexts that combine intuitionistic and linear assumptions, is as follows.

Simple Kinds	κ	$::=$	$\mathsf{t}^- \mid \tau \to \kappa$
Simple Types	τ	$::=$	$\alpha \mid \tau_1 \to \tau_2 \mid \tau_1 \multimap \tau_2 \mid \tau_1 \,\&\, \tau_2 \mid \top$
Simple Contexts	Σ	$::=$	$\epsilon \mid \Sigma, u{:}\tau$

$\Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau$	Type-Directed Object Equality
$\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau$	Structural Object Equality
$\Sigma \vdash A_1 \Longleftrightarrow A_2 : \kappa$	Kind-Directed Family Equality
$\Sigma \vdash A_1 \longleftrightarrow A_2 : \kappa$	Structural Family Equality
$\Sigma \vdash K_1 \longleftrightarrow K_2 : \mathsf{kind}^-$	Algorithmic Kind Equality

Figure 1: Algorithmic Equality Judgment Forms

5.1.2 Erasing Kinds, Types and Contexts

The erasure function $(\cdot)^-$ maps ordinary families, kinds and contexts to simple ones.

$$\begin{array}{rcl@{\qquad\qquad}rcl}
(a)^- & = & \alpha & (\mathsf{Type})^- & = & \mathsf{t}^- \\
(\lambda u{:}A_1.A_2)^- & = & A_2^- & (\Pi u{:}A.K)^- & = & A^- \to K^- \\
(A\,M)^- & = & A^- & & & \\
(\Pi u{:}A_1.A_2)^- & = & A_1^- \to A_2^- & (\epsilon)^- & = & \epsilon \\
(A_1 \multimap A_2)^- & = & A_1^- \multimap A_2^- & (\Gamma, u{:}A)^- & = & \Gamma^-, u{:}A^- \\
(A_1 \,\&\, A_2)^- & = & A_1^- \,\&\, A_2^- & (\Delta, u\hat{:}A)^- & = & \Delta^-, u{:}A^- \\
(\top)^- & = & \top & & &
\end{array}$$

To validate our intuition that erasure should remove all dependencies on terms, we prove the following lemma which states that substitutions into a family do not affect its erasure.

Lemma 23 (Erasure Preservation: Substitution)
For any family A, variable u and object M, $(A[M/u])^- = (A)^-$.

Proof: By structural induction on A.

■

Another useful feature of erasure is that definitionally equal families and kinds have identical erasures.

Lemma 24 (Erasure Preservation: Equality)

1. *If $\Gamma \vdash A = B : K$, then $A^- = B^-$.*
2. *If $\Gamma \vdash K = L : \mathsf{kind}$, then $K^- = L^-$.*

Proof: By induction on the equality derivation.

■

5.2 The Equality Algorithm

Our algorithmic equality judgment forms are shown in Figure 1. For objects and families we give both classifier-directed rules (that is, type-directed rules for comparing objects and kind-directed rules for comparing families) and structural rules. The classifier-directed rules apply extensionality until a base classifier is reached, then reduce to weak head normal form and compare structurally. The structural rules compare the constant, variable or primitive head and revert to the classifier-directed phase of the algorithm for any other subterms. Since there are no classifiers for kinds (or, put another way, every kind is of the same sort), we only need structural rules to compare kinds.

Intuitively, the classifier-directed portion of the algorithm takes a context, two terms, and a classifier and attempts to derive the corresponding algorithmic equality judgment, returning either success or failure; the structural portion takes a context and two terms in weak-head normal form and attempts to synthesize a simple type or kind for which the structural equality judgment is derivable, returning that classifier if it exists.

Notice that the algorithm ignores all issues of linearity. There is no distinction between intuitionistic and linear assumptions—the algorithm does not enforce any restrictions on the number of times something may be used—and the structural rule for linear applications does not split the context. Later, we will see that in the soundness proof, all the necessary information about allocation of linear assumptions is extracted from the typing derivations rather than the derivations of algorithmic equality.

Type-Directed Object Equality ($\Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau$)

$$\frac{M \xrightarrow{\text{wh}} M' \quad \Sigma \vdash M' \Longleftrightarrow N : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha} \qquad \frac{N \xrightarrow{\text{wh}} N' \quad \Sigma \vdash M \Longleftrightarrow N' : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha} \qquad \frac{\Sigma \vdash M \longleftrightarrow N : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha}$$

$$\frac{\Sigma, u{:}\tau_1 \vdash M\,u \Longleftrightarrow N\,u : \tau_2}{\Sigma \vdash M \Longleftrightarrow N : \tau_1 \to \tau_2} \qquad \frac{\Sigma, u{:}\tau_1 \vdash M\hat{\ }u \Longleftrightarrow N\hat{\ }u : \tau_2}{\Sigma \vdash M \Longleftrightarrow N : \tau_1 \multimap \tau_2} \qquad \frac{}{\Sigma \vdash M \Longleftrightarrow N : \top}$$

$$\frac{\Sigma \vdash \pi_1 M \Longleftrightarrow \pi_1 N : \tau_1 \quad \Sigma \vdash \pi_2 M \Longleftrightarrow \pi_2 N : \tau_2}{\Sigma \vdash M \Longleftrightarrow N : \tau_1 \mathbin{\&} \tau_2}$$

Structural Object Equality ($\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau$)

$$\frac{}{\Sigma \vdash u \longleftrightarrow u : \tau}\ (\Sigma(u) = \tau) \qquad \frac{}{\Sigma \vdash c \longleftrightarrow c : A^-}\ (S(c) = A) \qquad \frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_1 \mathbin{\&} \tau_2}{\Sigma \vdash \pi_i M_1 \longleftrightarrow \pi_i M_2 : \tau_i}$$

$$\frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_2 \to \tau_1 \quad \Sigma \vdash N_1 \Longleftrightarrow N_2 : \tau_2}{\Sigma \vdash M_1\,N_1 \longleftrightarrow M_2\,N_2 : \tau_1} \qquad \frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_2 \multimap \tau_1 \quad \Sigma \vdash N_1 \Longleftrightarrow N_2 : \tau_2}{\Sigma \vdash M_1\hat{\ }N_1 \longleftrightarrow M_2\hat{\ }N_2 : \tau_1}$$

Kind-Directed Family Equality ($\Sigma \vdash A_1 \Longleftrightarrow A_2 : \kappa$)

$$\frac{A \xrightarrow{\text{wh}} A' \quad \Sigma \vdash A' \Longleftrightarrow B : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-} \qquad \frac{B \xrightarrow{\text{wh}} B' \quad \Sigma \vdash A \Longleftrightarrow B' : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-}$$

$$\frac{\Sigma \vdash A \longleftrightarrow B : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-} \qquad \frac{\Sigma, u{:}\tau \vdash A\,u \Longleftrightarrow B\,u : \kappa}{\Sigma \vdash A \Longleftrightarrow B : \tau \to \kappa}$$

Structural Family Equality ($\Sigma \vdash A_1 \longleftrightarrow A_2 : \kappa$)

$$\frac{}{\Sigma \vdash a \longleftrightarrow a : K^-}\ (S(a) = K) \qquad \frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Sigma \vdash A_1 \multimap B_1 \longleftrightarrow A_2 \multimap B_2 : \mathsf{t}^-}$$

$$\frac{\Sigma \vdash A_1 \longleftrightarrow A_2 : \tau \to \kappa \quad \Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau}{\Sigma \vdash A_1\,M_1 \longleftrightarrow A_2\,M_2 : \kappa} \qquad \frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Sigma \vdash A_1 \mathbin{\&} B_1 \longleftrightarrow A_2 \mathbin{\&} B_2 : \mathsf{t}^-}$$

$$\frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma, u{:}A_1^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Sigma \vdash \Pi u{:}A_1.B_1 \longleftrightarrow \Pi u{:}A_2.B_2 : \mathsf{t}^-} \qquad \frac{}{\Sigma \vdash \top \longleftrightarrow \top : \mathsf{t}^-}$$

Structural Kind Equality ($\Sigma \vdash K_1 \longleftrightarrow K_2 : \mathsf{kind}^-$)

$$\frac{}{\Sigma \vdash \mathsf{t}^- \longleftrightarrow \mathsf{t}^- : \mathsf{kind}^-} \qquad \frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma, u{:}A_1^- \vdash K_1 \longleftrightarrow K_2 : \mathsf{kind}^-}{\Sigma \vdash \Pi u{:}A_1.K_1 \longleftrightarrow \Pi u{:}A_2.K_2 : \mathsf{kind}^-}$$

5.3 Some Properties of Algorithmic Equality

There are some elementary properties of the algorithmic equality judgments that may be proved immediately. We begin with weakening, which is easy to prove for the algorithmic judgments since they do not pay attention to linearity.

Lemma 25 (Weakening for Algorithmic Equality) *Let J be any algorithmic equality judgment. If $\Sigma, \Sigma' \vdash J$ then $\Sigma, u{:}\tau, \Sigma' \vdash J$.*

Proof: Straightforward, by induction on derivations.

■

The division of the algorithm into classifier-directed and structural rules constrains the structure of algorithmic equality derivations in a useful way. In particular, only weak head normal forms can be equated by the structural rules, and the context and terms in the structural judgments uniquely determine the classifier. These properties are formalized in the following lemma, which we call Determinacy because it essentially guarantees that no non-deterministic choices need ever be made in a bottom-up search for algorithmic equality derivations. For that reason, the lemma will play a role in proving decidability of equality, but we need it first to establish transitivity.

Lemma 26 (Determinacy of Algorithmic Equality)

1. *If $\Sigma \vdash M \longleftrightarrow N : \tau$ then there is no M' such that $M \xrightarrow{\text{wh}} M'$.*
2. *If $\Sigma \vdash M \longleftrightarrow N : \tau$ then there is no N' such that $N \xrightarrow{\text{wh}} N'$.*
3. *If $\Sigma \vdash A \longleftrightarrow B : \kappa$ then there is no A' such that $A \xrightarrow{\text{wh}} A'$.*
4. *If $\Sigma \vdash A \longleftrightarrow B : \kappa$ then there is no B' such that $B \xrightarrow{\text{wh}} B'$.*
5. *If $\Sigma \vdash M \longleftrightarrow N : \tau$ and $\Sigma \vdash M \longleftrightarrow N' : \tau'$ then $\tau = \tau'$.*
6. *If $\Sigma \vdash A \longleftrightarrow B : \kappa$ and $\Sigma \vdash A \longleftrightarrow B' : \kappa'$ then $\kappa = \kappa'$.*

Proof: By induction on the given derivations.

■

In order to prove the algorithmic equality relation is symmetric, we need another erasure preservation result in addition to Lemmas 23 and 24: the following lemma states that *algorithmically* equal families have the same erasure. Later, this will appear to be a trivial consequence of soundness and Lemma 24, but it is not difficult to prove directly. With this lemma established, the proof of symmetry is not complicated. The transitivity proof requires careful case analysis on the two derivations, using determinacy to rule out some cases that are impossible.

Lemma 27 (Erasure Preservation: Algorithmic Equality)

1. *If $\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-$, then $A^- = B^-$.*
2. *If $\Sigma \vdash A \longleftrightarrow B : \kappa$, then $A^- = B^-$.*
3. *If $A \xrightarrow{\text{wh}} B$, then $A^- = B^-$.*

Proof: By induction on the given derivation.

■

Lemma 28 (Symmetry of Algorithmic Equality)

1. *If $\Sigma \vdash M \Longleftrightarrow N : \tau$ then $\Sigma \vdash N \Longleftrightarrow M : \tau$.*
2. *If $\Sigma \vdash M \longleftrightarrow N : \tau$ then $\Sigma \vdash N \longleftrightarrow M : \tau$.*

3. *If* $\Sigma \vdash A \Longleftrightarrow B : \kappa$ *then* $\Sigma \vdash B \Longleftrightarrow A : \kappa$.

4. *If* $\Sigma \vdash A \longleftrightarrow B : \kappa$ *then* $\Sigma \vdash B \longleftrightarrow A : \kappa$.

5. *If* $\Sigma \vdash K \longleftrightarrow L : \mathsf{kind}^-$ *then* $\Sigma \vdash L \longleftrightarrow K : \mathsf{kind}^-$.

Proof: By induction on the given derivations. The proof is entirely straightforward, except that Lemma 27 is required for the cases of structural comparison of dependent products. The case for product kinds is similar to that for product families, so we will only show the latter.

Case:

$$\frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma, u{:}A_1^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Sigma \vdash \Pi u{:}A_1.B_1 \longleftrightarrow \Pi u{:}A_2.B_2 : \mathsf{t}^-}$$

By the i.h. on the first subderivation, $\Sigma \vdash A_2 \Longleftrightarrow A_1 : \mathsf{t}^-$.
By the i.h. on the second subderivation, $\Sigma, u{:}A_1^- \vdash B_2 \Longleftrightarrow B_1 : \mathsf{t}^-$.
By Lemma 27, $A_1^- = A_2^-$, so $\Sigma, u{:}A_2^- \vdash B_2 \Longleftrightarrow B_1 : \mathsf{t}^-$.
By the same rule, $\Sigma \vdash \Pi u{:}A_2.B_2 \longleftrightarrow \Pi u{:}A_1.B_1 : \mathsf{t}^-$.

■

Lemma 29 (Transitivity of Algorithmic Equality)

1. *If* $\Sigma \vdash M \Longleftrightarrow N : \tau$ *and* $\Sigma \vdash N \Longleftrightarrow O : \tau$, *then* $\Sigma \vdash M \Longleftrightarrow O : \tau$.

2. *If* $\Sigma \vdash M \longleftrightarrow N : \tau$ *and* $\Sigma \vdash N \longleftrightarrow O : \tau$, *then* $\Sigma \vdash M \longleftrightarrow O : \tau$.

3. *If* $\Sigma \vdash A \Longleftrightarrow B : \kappa$ *and* $\Sigma \vdash B \Longleftrightarrow C : \kappa$, *then* $\Sigma \vdash A \Longleftrightarrow C : \kappa$.

4. *If* $\Sigma \vdash A \longleftrightarrow B : \kappa$ *and* $\Sigma \vdash B \longleftrightarrow C : \kappa$, *then* $\Sigma \vdash A \longleftrightarrow C : \kappa$.

5. *If* $\Sigma \vdash K \longleftrightarrow L : \mathsf{kind}^-$ *and* $\Sigma \vdash L \longleftrightarrow L' : \mathsf{kind}^-$, *then* $\Sigma \vdash K \longleftrightarrow L' : \mathsf{kind}^-$.

Proof: By simultaneous induction on the two given derivations, using determinacy (Lemmas 15 and 26). We will show several cases; the others are all straightforward.

Case: The first derivation ends with

$$\frac{M \xrightarrow{\text{wh}} M' \quad \Sigma \vdash M' \Longleftrightarrow N : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha}$$

By the i.h., $\Sigma \vdash M' \Longleftrightarrow O : \alpha$.
By the same rule, $\Sigma \vdash M \Longleftrightarrow O : \alpha$.

Case: The second derivation ends with

$$\frac{O \xrightarrow{\text{wh}} O' \quad \Sigma \vdash N \Longleftrightarrow O' : \alpha}{\Sigma \vdash N \Longleftrightarrow O : \alpha}$$

By the i.h., $\Sigma \vdash M \Longleftrightarrow O' : \alpha$.
By the same rule, $\Sigma \vdash M \Longleftrightarrow O : \alpha$.

Case:

$$\frac{N \xrightarrow{\text{wh}} N' \quad \Sigma \vdash M \Longleftrightarrow N' : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha} \quad \text{and} \quad \frac{N \xrightarrow{\text{wh}} N'' \quad \Sigma \vdash N'' \Longleftrightarrow O : \alpha}{\Sigma \vdash N \Longleftrightarrow O : \alpha}$$

By Lemma 15, part (1), $N' = N''$.
Thus, we may apply the i.h. and get $\Sigma \vdash M \Longleftrightarrow N : \alpha$.

Case:

$$\frac{N \xrightarrow{\text{wh}} N' \quad \Sigma \vdash M \Longleftrightarrow N' : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha} \quad \text{and} \quad \frac{\Sigma \vdash N \longleftrightarrow O : \alpha}{\Sigma \vdash N \Longleftrightarrow O : \alpha}$$

This case is impossible by Lemma 26, part (1).
The symmetric case is impossible by part (2) of the same lemma.

Case:

$$\frac{\Sigma \vdash M' \longleftrightarrow N' : \tau_2 \to \tau_1 \quad \Sigma \vdash M'' \Longleftrightarrow N'' : \tau_2}{\Sigma \vdash M'\, M'' \longleftrightarrow N'\, N'' : \tau_1}$$

$$\text{and} \quad \frac{\Sigma \vdash N' \longleftrightarrow O' : \tau_3 \to \tau_1 \quad \Sigma \vdash N'' \Longleftrightarrow O'' : \tau_3}{\Sigma \vdash N'\, N'' \longleftrightarrow O'\, O'' : \tau_1}$$

By Lemmas 28 and 26, $\tau_2 = \tau_3$.
By the i.h., $\Sigma \vdash M' \longleftrightarrow O' : \tau_2 \to \tau_1$ and $\Sigma \vdash M'' \Longleftrightarrow O'' : \tau_2$.
By the same rule, $\Sigma \vdash M'\, M'' \longleftrightarrow O'\, O'' : \tau_1$.

Case:

$$\frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_2 \multimap \tau_1 \quad \Sigma \vdash N_1 \Longleftrightarrow N_2 : \tau_2}{\Sigma \vdash M_1\hat{\ }N_1 \longleftrightarrow M_2\hat{\ }N_2 : \tau_1}$$

$$\text{and} \quad \frac{\Sigma \vdash M_2 \longleftrightarrow M_3 : \tau_2' \multimap \tau_1 \quad \Sigma \vdash N_2 \Longleftrightarrow N_3 : \tau_2'}{\Sigma \vdash M_2\hat{\ }N_2 \longleftrightarrow M_3\hat{\ }N_3 : \tau_1}$$

Similar to the previous case.

Case:

$$\frac{\Sigma \vdash M' \longleftrightarrow N' : \tau_1 \,\&\, \tau_2}{\Sigma \vdash \pi_1 M' \longleftrightarrow \pi_1 N' : \tau_1} \quad \text{and} \quad \frac{\Sigma \vdash N' \longleftrightarrow O' : \tau_1 \,\&\, \tau_2'}{\Sigma \vdash \pi_1 N' \longleftrightarrow \pi_1 O' : \tau_1}$$

By Lemmas 28 and 26, $\tau_2 = \tau_2'$.
By the i.h., $\Sigma \vdash M' \longleftrightarrow O' : \tau_1 \,\&\, \tau_2$.
By the same rule, $\Sigma \vdash \pi_1 M' \longleftrightarrow \pi_1 O' : \tau_1$

Case:

$$\frac{\Sigma \vdash M' \longleftrightarrow N' : \tau_1 \,\&\, \tau_2}{\Sigma \vdash \pi_2 M' \longleftrightarrow \pi_2 N' : \tau_2} \quad \text{and} \quad \frac{\Sigma \vdash N' \longleftrightarrow O' : \tau_1' \,\&\, \tau_2}{\Sigma \vdash \pi_2 N' \longleftrightarrow \pi_2 O' : \tau_2}$$

Similar to the previous case.

Case: The first derivation ends with

$$\frac{A \xrightarrow{\text{wh}} A' \quad \Sigma \vdash A' \Longleftrightarrow B : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-}$$

By the i.h., $\Sigma \vdash A' \Longleftrightarrow C : \mathsf{t}^-$.
By the same rule, $\Sigma \vdash A \Longleftrightarrow C : \mathsf{t}^-$.

Case: The second derivation ends with

$$\frac{C \xrightarrow{\text{wh}} C' \quad \Sigma \vdash B \Longleftrightarrow C' : \mathsf{t}^-}{\Sigma \vdash B \Longleftrightarrow C : \mathsf{t}^-}$$

Similar to the previous case.

Case:

$$\frac{B \xrightarrow{\text{wh}} B' \quad \Sigma \vdash A \Longleftrightarrow B' : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-} \quad \text{and} \quad \frac{B \xrightarrow{\text{wh}} B'' \quad \Sigma \vdash B'' \Longleftrightarrow C : \mathsf{t}^-}{\Sigma \vdash B \Longleftrightarrow C : \mathsf{t}^-}$$

By Lemma 15, part (2), $B' = B''$.
Thus, we may apply the i.h. and get $\Sigma \vdash A \Longleftrightarrow C : \mathsf{t}^-$.

Case:

$$\frac{B \xrightarrow{\text{wh}} B' \quad \Sigma \vdash A \Longleftrightarrow B' : \mathsf{t}^-}{\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-} \quad \text{and} \quad \frac{\Sigma \vdash B \longleftrightarrow C : \mathsf{t}^-}{\Sigma \vdash B \Longleftrightarrow C : \mathsf{t}^-}$$

This case is impossible by Lemma 26, part (4).
The symmetric case is impossible by part (5) of the same lemma.

Case:

$$\frac{\Sigma \vdash A_1 \longleftrightarrow A_2 : \tau \to \kappa \quad \Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau}{\Sigma \vdash A_1\, M_1 \longleftrightarrow A_2\, M_2 : \kappa} \quad \text{and} \quad \frac{\Sigma \vdash A_2 \longleftrightarrow A_3 : \tau' \to \kappa \quad \Sigma \vdash M_2 \Longleftrightarrow M_3 : \tau'}{\Sigma \vdash A_2\, M_2 \longleftrightarrow A_3\, M_3 : \kappa}$$

By Lemmas 28 and 26, $\tau = \tau'$.
By the i.h., $\Sigma \vdash A_1 \longleftrightarrow A_3 : \tau \to \kappa$ and $\Sigma \vdash M_1 \Longleftrightarrow M_3 : \tau$.
By the same rule, $\Sigma \vdash A_1\, M_1 \longleftrightarrow A_3\, M_3 : \kappa$.

Case:

$$\frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma, u{:}A_1^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Sigma \vdash \Pi u{:}A_1.B_1 \longleftrightarrow \Pi u{:}A_2.B_2 : \mathsf{t}^-} \quad \text{and} \quad \frac{\Sigma \vdash A_2 \Longleftrightarrow A_3 : \mathsf{t}^- \quad \Sigma, u{:}A_2^- \vdash B_2 \Longleftrightarrow B_3 : \mathsf{t}^-}{\Sigma \vdash \Pi u{:}A_2.B_2 \longleftrightarrow \Pi u{:}A_3.B_3 : \mathsf{t}^-}$$

By Lemma 27, $A_1^- = A_2^-$.
By the i.h., $\Sigma \vdash A_1 \Longleftrightarrow A_3 : \mathsf{t}^-$ and $\Sigma, u{:}A_1^- \vdash B_1 \Longleftrightarrow B_3 : \mathsf{t}^-$.
By the same rule, $\Sigma \vdash \Pi u{:}A_1.B_1 \longleftrightarrow \Pi u{:}A_3.B_3 : \mathsf{t}^-$.

Case:

$$\frac{\Sigma \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Sigma, u{:}A_1^- \vdash K_1 \longleftrightarrow K_2 : \mathsf{kind}^-}{\Sigma \vdash \Pi u{:}A_1.K_1 \longleftrightarrow \Pi u{:}A_2.K_2 : \mathsf{kind}^-}$$

$$\text{and} \quad \frac{\Sigma \vdash A_2 \Longleftrightarrow A_3 : \mathsf{t}^- \quad \Sigma, u{:}A_2^- \vdash K_2 \longleftrightarrow K_3 : \mathsf{kind}^-}{\Sigma \vdash \Pi u{:}A_2.K_2 \longleftrightarrow \Pi u{:}A_3.K_3 : \mathsf{kind}^-}$$

By Lemma 27, $A_1^- = A_2^-$.
By the i.h., $\Sigma \vdash A_1 \Longleftrightarrow A_3 : \mathsf{t}^-$ and $\Sigma, u{:}A_1^- \vdash K_1 \longleftrightarrow K_3 : \mathsf{kind}^-$.
By the same rule, $\Sigma \vdash \Pi u{:}A_1.K_1 \longleftrightarrow \Pi u{:}A_3.K_3 : \mathsf{kind}^-$

■

6 Soundness of Algorithmic Equality

In this section we will prove the soundness result for our algorithm; essentially, we want to show that if two terms are algorithmically equal then they are definitionally equal. It is clear, however, that this can only be true for well-typed terms. (Consider the type-directed rule at type $\top$!) Our soundness theorem will therefore have to require that typing derivations exist for the terms being compared. Since our algorithm does not enforce the linearity restrictions present in the definitional equality rules, the proof must also rely on the typing derivations to determine how linear contexts should be split among premises when dealing with linear function applications.

This can pose some difficulty if the two typing derivations disagree on how the context should be split. This can't be avoided, as equal terms may sometimes use their resources differently if unit expressions are involved. For example, in the context $\Delta = u{:}\top, v{:}\top, w{:}\top \multimap \top \multimap A$, the terms $(w\,\hat{}\,\langle\rangle)\,\hat{}\,u$ and $(w\,\hat{}\,\langle\rangle)\,\hat{}\,v$ are equal, but there is no linear context in which u and v may be simultaneously well-typed, let alone equal.

To solve this problem, we follow a suggestion proposed by Pfenning [8]. The key is to observe that the way to prove those two problematic applications equal is to use the fact that any variable of type $\top$ is equal to $\langle\rangle$. Using this extensionality rule and congruence rules, we prove that both of the above terms are equal to $(w\,\hat{}\,\langle\rangle)\,\hat{}\,\langle\rangle$; thus by transitivity they are equal to each other. But changing an expression of type $\top$ into $\langle\rangle$ is just η-expansion, and HP showed that the type-directed algorithm can be instrumented to find η-long forms. Therefore, the soundness proof should, rather than directly proving the algorithmically equal terms to be definitionally equal, extract a mediating term and prove that it is definitionally equal to both. Comparison with HP's discussion of pseudo-canonical forms strongly suggests that in the classifier-directed cases of the proof, this mediating term will be canonical except for the type labels on λ-abstractions, but we will not prove this.

Before we tackle the main soundness theorem, we must prove a subject reduction lemma. We have expressed this property a little differently from HP, but Regularity (Lemma 10) ensures these two formulations are equivalent provided all the contexts involved are valid.

Lemma 30 (Subject Reduction)

1. *If* $\vdash \Gamma\ \mathsf{context}$ *and* $\Gamma \vdash A : K$ *and* $A \xrightarrow{\text{wh}} A'$, *then* $\Gamma \vdash A = A' : K$.
2. *If* $\vdash \Gamma\ \mathsf{context}$ *and* $\Gamma \vdash \Delta\ \mathsf{context}$ *and* $\Gamma; \Delta \vdash M : A$ *and* $M \xrightarrow{\text{wh}} M'$, *then* $\Gamma; \Delta \vdash M = M' : A$.

Proof: By induction on weak head reduction derivations. Injectivity (Theorem 1) is required in the cases for the β-reduction axioms for projections and applications. We will show a few representative cases; the rest are analogous.

Case:

$$\frac{}{(\hat{\lambda} u{:}B.M_1)\,\hat{}\,M_2 \xrightarrow{\text{wh}} M_1[M_2/u]}$$

By Lemma 13, $\Delta = (\Delta_1, \Delta_2)$ where $\Gamma; \Delta_1 \vdash \hat{\lambda} u{:}B.M_1 : A_1 \multimap A_2$ and $\Gamma; \Delta_2 \vdash M_2 : A_1$ and $\Gamma \vdash A_2 = A : \mathsf{Type}$.
By Lemma 13, $\Gamma \vdash A_1 \multimap A_2 = B \multimap A_2' : \mathsf{Type}$ and $\Gamma \vdash B : \mathsf{Type}$ and $\Gamma; \Delta_1, u\hat{:}B \vdash M_1 : A_2'$.
By injectivity, $\Gamma \vdash A_1 = B : \mathsf{Type}$ and $\Gamma \vdash A_2 = A_2' : \mathsf{Type}$.
By type conversion, $\Gamma; \Delta_2 \vdash M_2 : B$.
By reflexivity, $\Gamma; \Delta_2 \vdash M_2 = M_2 : B$ and $\Gamma; \Delta_1, u\hat{:}B \vdash M_1 = M_1 : A_2'$.
By parallel conversion, $\Gamma; \Delta \vdash (\hat{\lambda} u{:}B.M_1)\,\hat{}\,M_2 = M_1[M_2/u] : A_2'$.
By symmetry and transitivity, $\Gamma \vdash A_2' = A : \mathsf{Type}$.
By type conversion, $\Gamma; \Delta \vdash (\hat{\lambda} u{:}B.M_1)\,\hat{}\,M_2 = M_1[M_2/u] : A$.

Case:

$$\frac{M_1 \xrightarrow{\text{wh}} M_1'}{M_1\,\hat{}\,M_2 \xrightarrow{\text{wh}} M_1'\,\hat{}\,M_2}$$

By Lemma 13, $\Delta = (\Delta_1, \Delta_2)$ where $\Gamma; \Delta_1 \vdash M_1 : A_1 \multimap A_2$ and $\Gamma; \Delta_2 \vdash M_2 : A_1$ and $\Gamma \vdash A_2 = A : \mathsf{Type}$.
By the i.h., $\Gamma; \Delta_1 \vdash M_1 = M_1' : A_1 \multimap A_2$.
By reflexivity, $\Gamma; \Delta_2 \vdash M_2 = M_2 : A_1$.
By congruence, $\Gamma; \Delta \vdash M_1\,\hat{}\,M_2 = M_1'\,\hat{}\,M_2 : A_2$.
By type conversion, $\Gamma; \Delta \vdash M_1\,\hat{}\,M_2 = M_1'\,\hat{}\,M_2 : A$.

Case:

$$\frac{}{(\lambda u{:}A_1.A_2)\,M \xrightarrow{\text{wh}} A_2[M/u]}$$

By Lemma 13, $\Gamma \vdash \lambda u{:}A_1.A_2 : \Pi u{:}B_1.K_1$ and $\Gamma; \epsilon \vdash M : B_1$ and $\Gamma \vdash K_1[M/u] = K : \mathsf{kind}$.
By Lemma 13, $\Gamma \vdash \Pi u{:}B_1.K_1 = \Pi u{:}A_1.K_1' : \mathsf{kind}$ and $\Gamma \vdash A_1 : \mathsf{Type}$ and $\Gamma, u{:}A_1 \vdash A_2 : K_1'$.

By Lemma 14, $\Gamma \vdash B_1 = A_1 : \mathsf{Type}$ and $\Gamma, u{:}B_1 \vdash K_1 = K_1' : \mathsf{kind}$.
By type conversion, $\Gamma; \epsilon \vdash M : A_1$.
By reflexivity, $\Gamma; \epsilon \vdash M = M : A_1$ and $\Gamma, u{:}A_1 \vdash A_2 = A_2 : K_1'$.
By parallel conversion, $\Gamma \vdash (\lambda u{:}A_1.A_2)\, M = A_2[M/u] : K_1'[M/u]$.
By reflexivity, $\Gamma; \epsilon \vdash M = M : B_1$.
By substitution, $\Gamma \vdash K_1[M/u] = K_1'[M/u] : \mathsf{kind}$.
By symmetry and transitivity, $\Gamma \vdash K_1'[M/u] = K : \mathsf{kind}$.
By type conversion, $\Gamma \vdash (\lambda u{:}A_1.A_2)\, M = A_2[M/u] : K$.

Case:

$$\frac{A \xrightarrow{\text{wh}} A'}{A\, M \xrightarrow{\text{wh}} A'\, M}$$

By Lemma 13, $\Gamma \vdash A : \Pi u{:}B_1.K_1$ and $\Gamma; \epsilon \vdash M : B_1$ and $\Gamma \vdash K_1[M/u] = K : \mathsf{kind}$.
By the i.h., $\Gamma \vdash A = A' : \Pi u{:}B_1.K_1$.
By reflexivity, $\Gamma; \epsilon \vdash M = M : B_1$.
By congruence, $\Gamma \vdash A\, M = A'\, M : K_1[M/u]$.
By type conversion, $\Gamma \vdash A\, M = B\, M : K$.

■

Now we can prove the main lemma of this section, which will imply soundness of the algorithm. Given two terms that are well-formed and algorithmically equal, the proof constructs a term that is definitionally equal to each of them. Decisions about how the linear context should be split in the definitional equality derivations are made based on the given typing derivations, and since two separate equality derivations are being constructed, there is no need to attempt to resolve differences between the two typing derivations. Soundness of algorithmic equality follows directly from this lemma, using transitivity.

Lemma 31 (Well-Formed Terms that are Algorithmically Equal have Mediating Terms) *Assume* $\vdash \Gamma$ $\mathsf{context}$ *and, where applicable,* $\Gamma \vdash \Delta_i$ $\mathsf{context}$. *Assume further that if* $(u\hat{:}C) \in \Delta_1$ *and* $(u\hat{:}C') \in \Delta_2$*), then* $C = C'$.

1. *If* $\Gamma; \Delta_1 \vdash M : A$ *and* $\Gamma; \Delta_2 \vdash N : A$ *and* $\Sigma \vdash M \Longleftrightarrow N : A^-$ *and* $\Gamma^-, \Delta_i^- \subseteq \Sigma$ *for each* i*, then there is some* P *such that* $\Gamma; \Delta_1 \vdash P = M : A$ *and* $\Gamma; \Delta_2 \vdash P = N : A$.

2. *If* $\Gamma; \Delta_1 \vdash M : A$ *and* $\Gamma; \Delta_2 \vdash N : B$ *and* $\Sigma \vdash M \longleftrightarrow N : \tau$ *and* $\Gamma^-, \Delta_i^- \subseteq \Sigma$ *for each* i*, then* $\Gamma \vdash A = B : \mathsf{Type}$ *and there is some* P *such that* $\Gamma; \Delta_1 \vdash P = M : A$ *and* $\Gamma; \Delta_2 \vdash P = N : A$ *and* $A^- = B^- = \tau$.

3. *If* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : K$ *and* $\Gamma^- \vdash A \Longleftrightarrow B : K^-$*, then* $\Gamma \vdash A = B : K$.

4. *If* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : L$ *and* $\Gamma \vdash A \longleftrightarrow B : \kappa$*, then* $\Gamma \vdash A = B : K$ *and* $\Gamma \vdash K = L : \mathsf{kind}$ *and* $K^- = L^- = \kappa$.

5. *If* $\Gamma \vdash K : \mathsf{kind}$ *and* $\Gamma \vdash L : \mathsf{kind}$ *and* $\Gamma^- \vdash K \longleftrightarrow L : \mathsf{kind}^-$ *then* $\Gamma \vdash K = L : \mathsf{kind}$.

Proof: By induction on the algorithmic equality derivation, using inversion and injectivity properties.

Case:

$$\frac{M \xrightarrow{\text{wh}} M' \quad \Sigma \vdash M' \Longleftrightarrow N : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha}$$

By Lemma 30, $\Gamma; \Delta_1 \vdash M = M' : A$.
By Lemma 10, $\Gamma; \Delta_1 \vdash M' : A$.
By the i.h., $\Gamma; \Delta_1 \vdash P = M' : A$ and $\Gamma; \Delta_2 \vdash P = N : B$.
By symmetry and transitivity, $\Gamma; \Delta_1 \vdash P = M : A$.

Case:

$$\frac{N \xrightarrow{\text{wh}} N' \quad \Sigma \vdash M \Longleftrightarrow N' : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha}$$

Similar to the previous case.

Case:

$$\frac{\Sigma \vdash M \longleftrightarrow N : \alpha}{\Sigma \vdash M \Longleftrightarrow N : \alpha}$$

By the i.h., $\Gamma \vdash C = A : \mathsf{Type}$ and $\Gamma; \Delta_1 \vdash P = M : C$ and $\Gamma; \Delta_2 \vdash P = N : C$.
By type conversion, $\Gamma; \Delta_1 \vdash P = M : A$ and $\Gamma; \Delta_2 \vdash P = N : A$.

Case:

$$\frac{\Sigma, u{:}\tau_1 \vdash M_1\, u \Longleftrightarrow M_2\, u : \tau_2}{\Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau_1 \to \tau_2}$$

Since $A^- = \tau_1 \to \tau_2$, $A = \Pi u{:}A_1.A_2$ where $A_1^- = \tau_1$ and $A_2^- = \tau_2$.
By Lemmas 10 and 13, $\Gamma \vdash A_1 : \mathsf{Type}$, and so $\vdash \Gamma, u{:}A_1$ $\mathsf{context}$.
Thus by weakening and rules, $\Gamma, u{:}A_1 \vdash \Delta_i$ $\mathsf{context}$ for each i, and $\Gamma, u{:}A_1; \Delta_1 \vdash M_1\, u : A_2$
and $\Gamma, u{:}A_1; \Delta_2 \vdash N_1\, u : A_2$.
Also, note that $(\Gamma, u{:}A_1)^- \subseteq (\Sigma, u{:}\tau_1)$.
By the i.h., $\Gamma, u{:}A_1; \Delta_1 \vdash P = M_1\, u : A_2$ and $\Gamma, u{:}A_1; \Delta_2 \vdash P = M_2\, u : A_2$.
By symmetry and transitivity, $\Gamma, u{:}A_1; \Delta_i \vdash P = P : A_2$.
Let v be a fresh variable. By renaming, $\Gamma, v{:}A_1; \Delta_i \vdash P[v/u] = P[v/u] : A_2[v/u]$.
By weakening, $\Gamma, u{:}A_1, v{:}A_1; \Delta_i \vdash P[v/u] = P[v/u] : A_2[v/u]$.
By rule, $\Gamma, u{:}A_1; \epsilon \vdash u = u : A_1$.
By parallel conversion, $\Gamma, u{:}A_1; \Delta_i \vdash (\lambda v{:}A_1.P[v/u])\, u = P[v/u][u/v] : A_2$.
By transitivity, $\Gamma, u{:}A_1; \Delta_i \vdash (\lambda v{:}A_1.P[v/u])\, u = M_i\, u : A_2$.
By extensionality (and renaming), $\Gamma; \Delta_i \vdash \lambda u{:}A_1.P = M_i : \Pi u{:}A_1.A_2$.

Case:

$$\frac{\Sigma, u\hat{:}\tau_1 \vdash M_1\,\hat{}\,u \Longleftrightarrow M_2\,\hat{}\,u : \tau_2}{\Sigma \vdash M_1 \Longleftrightarrow M_2 : \tau_1 \multimap \tau_2}$$

Since $A^- = \tau_1 \multimap \tau_2$, $A = A_1 \multimap A_2$ where $A_1^- = \tau_1$ and $A_2^- = \tau_2$.
By Lemmas 10 and 13, $\Gamma \vdash A_1 : \mathsf{Type}$ and so $\Gamma \vdash \Delta_i, u\hat{:}A_1$ $\mathsf{context}$ for each i.
By rules, $\Gamma; \Delta_1, u\hat{:}A_1 \vdash M_1\,\hat{}\,u : A_2$ and $\Gamma; \Delta_2, u\hat{:}A_1 \vdash M_2\,\hat{}\,u : A_2$.
Also, note that $(\Delta_i, u\hat{:}A_1)^- \subseteq (\Sigma, u{:}\tau_1)$.
By the i.h., $\Gamma; \Delta_1, u\hat{:}A_1 \vdash P = M_1\,\hat{}\,u : A_2$ and $\Gamma; \Delta_2, u\hat{:}A_1 \vdash P = M_2\,\hat{}\,u : A_2$.
By symmetry and transitivity, $\Gamma; \Delta_i, u\hat{:}A_1 \vdash P = P : A_2$.
By rule, $\Gamma; u\hat{:}A_1 \vdash u = u : A_1$.
By parallel conversion, $\Gamma; \Delta_i, u\hat{:}A_1 \vdash (\hat{\lambda} u{:}A_1.P)\,\hat{}\,u = P : A_2$.
By transitivity, $\Gamma; \Delta_i, u\hat{:}A_1 \vdash (\hat{\lambda} u{:}A_1.P)\,\hat{}\,u = M_i\,\hat{}\,u : A_2$.
By extensionality, $\Gamma; \Delta_i \vdash \hat{\lambda} u{:}A_1.P = M_i : A_1 \multimap A_2$.

Case:

$$\frac{\Sigma \vdash \pi_1 M \Longleftrightarrow \pi_1 N : \tau_1 \quad \Sigma \vdash \pi_2 M \Longleftrightarrow \pi_2 N : \tau_2}{\Sigma \vdash M \Longleftrightarrow N : \tau_1 \mathbin{\&} \tau_2}$$

Since $A^- = \tau_1 \mathbin{\&} \tau_2$, $A = A_1 \mathbin{\&} A_2$ where $A_1^- = \tau_1$ and $A_2^- = \tau_2$.
Thus by rules, $\Gamma; \Delta_1 \vdash \pi_i M : A_i$ and $\Gamma; \Delta_2 \vdash \pi_i N : A_i$, for $i = 1, 2$.
By the i.h., there exist P_1, P_2 such that $\Gamma; \Delta_1 \vdash P_1 = \pi_1 M : A_1$ and $\Gamma; \Delta_2 \vdash P_1 = \pi_1 N : A_1$ and $\Gamma; \Delta_1 \vdash P_2 = \pi_2 M : A_2$ and $\Gamma; \Delta_2 \vdash P_2 = \pi_2 N : A_2$.
By parallel conversion, $\Gamma; \Delta_1 \vdash \pi_i \langle P_1, P_2 \rangle = \pi_i M : A_1$ and $\Gamma; \Delta_2 \vdash \pi_i \langle P_1, P_2 \rangle = \pi_i N : A_2$.
By extensionality, $\Gamma; \Delta_1 \vdash \langle P_1, P_2 \rangle = M : A_1 \mathbin{\&} A_2$ and $\Gamma; \Delta_2 \vdash \langle P_1, P_2 \rangle = N : A_1 \mathbin{\&} A_2$.

Case:

$$\frac{}{\Sigma \vdash M \Longleftrightarrow N : \top}$$

Since $A^- = \top$, $A = \top$.
By rule, $\Gamma; \Delta_1 \vdash \langle\rangle : \top$ and $\Gamma; \Delta_2 \vdash \langle\rangle : \top$.
Thus by extensionality, $\Gamma; \Delta_1 \vdash \langle\rangle = M : \top$ and $\Gamma; \Delta_2 \vdash \langle\rangle = N : \top$.

Case:

$$\frac{}{\Sigma \vdash u \longleftrightarrow u : \tau}\ (\Sigma(u) = \tau)$$

Lemma 13 gives two subcases:

> **Sub-Case:** $\Delta_1 = \Delta_2 = \epsilon$, $\Gamma(u) = C$, $\Gamma \vdash C = A : \mathsf{Type}$ and $\Gamma \vdash C = B : \mathsf{Type}$.
> By symmetry and transitivity, $\Gamma \vdash A = B : \mathsf{Type}$. Using erasure preservation, $A^- = B^- = C^- = \tau$.
> Let $P = u$. By rule, $\Gamma; \Delta_i \vdash P = u : C$. By type conversion, $\Gamma; \Delta_i \vdash P = u : A$.
>
> **Sub-Case:** $\Delta_1 = \Delta_2 = (u \hat{:} C)$, $\Gamma \vdash C = A : \mathsf{Type}$ and and $\Gamma \vdash C = B : \mathsf{Type}$.
> By symmetry and transitivity, $\Gamma \vdash A = B : \mathsf{Type}$.
> Using erasure preservation, $A^- = B^- = C^- = \tau$.
> Let $P = u$. By rule, $\Gamma; \Delta_i \vdash P = u : C$.
> By type conversion, $\Gamma; \Delta_i \vdash P = u : A$.

Case:

$$\frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_2 \to \tau_1 \quad \Sigma \vdash N_1 \Longleftrightarrow N_2 : \tau_2}{\Sigma \vdash M_1\, N_1 \longleftrightarrow M_2\, N_2 : \tau_1}$$

By Lemma 13, $\Gamma; \Delta_1 \vdash M_1 : \Pi u{:}A_2.A_1$ and $\Gamma; \epsilon \vdash N_1 : A_2$ and $\Gamma \vdash A_1[N_1/u] = A : \mathsf{Type}$.
Similarly, $\Gamma; \Delta_2 \vdash M_2 : \Pi u{:}B_2.B_1$ and $\Gamma; \epsilon \vdash N_2 : B_2$ and $\Gamma \vdash B_1[N_2/u] = B : \mathsf{Type}$.
By the i.h. on the first subderivation, $\Gamma \vdash C = \Pi u{:}A_2.A_1 : \mathsf{Type}$ and $\Gamma \vdash C = \Pi u{:}B_2.B_1 :$ and $\Gamma; \Delta_1 \vdash P = M_1 : C$ and $\Gamma; \Delta_2 \vdash P = M_2 : C$.
Also (using the definition of erasure), $A_1^- = B_1^- = \tau_1$ and $A_2^- = B_2^- = \tau_2$.
By symmetry and transitivity, $\Gamma \vdash \Pi u{:}A_2.A_1 = \Pi u{:}B_2.B_1 : \mathsf{Type}$.
By injectivity, $\Gamma \vdash A_2 = B_2 : \mathsf{Type}$ and $\Gamma, u{:}A_2 \vdash A_1 = B_1 : \mathsf{Type}$.
By type conversion, $\Gamma; \epsilon \vdash N_2 : A_2$ and $\Gamma; \Delta_1 \vdash P = M_1 : \Pi u{:}A_2.A_1$ and $\Gamma; \Delta_2 \vdash P = M_2 : \Pi u{:}A_2.A_1$.
By the i.h. on the second subderivation, $\Gamma; \epsilon \vdash Q = N_1 : A_2$ and $\Gamma; \epsilon \vdash Q = N_2 : A_2$.
By functionality, $\Gamma \vdash A_1[Q/u] = A_1[N_1/u] : \mathsf{Type}$ and $\Gamma \vdash A_1[Q/u] = B_1[N_2/u] : \mathsf{Type}$.
By transitivity, $\Gamma \vdash A_1[Q/u] = A : \mathsf{Type}$ and $\Gamma \vdash A_1[Q/u] = B : \mathsf{Type}$.
By symmetry and transitivity, $\Gamma \vdash A = B : \mathsf{Type}$.
By a congruence rule, $\Gamma; \Delta_1 \vdash P\, Q = M_1\, N_1 : A_1[Q/u]$ and $\Gamma; \Delta_2 \vdash P\, Q = M_2\, N_2 : A_1[Q/u]$.
By type conversion, $\Gamma; \Delta_i \vdash P\, Q = M_i\, N_i : A$.
Finally, using erasure preservation, $A^- = B^- = \tau_1$.

Case:

$$\frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_2 \multimap \tau_1 \quad \Sigma \vdash N_1 \Longleftrightarrow N_2 : \tau_2}{\Sigma \vdash M_1 \hat{}\, N_1 \longleftrightarrow M_2 \hat{}\, N_2 : \tau_1}$$

By Lemma 13, $\Delta_1 = (\Delta_1', \Delta_1'')$ where $\Gamma; \Delta_1' \vdash M_1 : A_2 \multimap A_1$ and $\Gamma; \Delta_1'' \vdash N_1 : A_2$ and $\Gamma \vdash A_1 = A : \mathsf{Type}$.
Similarly, $\Delta_2 = (\Delta_2', \Delta_2'')$ where $\Gamma; \Delta_2' \vdash M_2 : B_2 \multimap B_1$ and $\Gamma; \Delta_2'' \vdash N_2 : B_2$ and $\Gamma \vdash B_1 = B : \mathsf{Type}$.
Observe that $\Gamma \vdash \Delta_i'$ $\mathsf{context}$ and $\Gamma \vdash \Delta_i''$ $\mathsf{context}$, for $i = 1, 2$.
By the i.h. on the first subderivation, $\Gamma \vdash C = A_2 \multimap A_1 : \mathsf{Type}$ and $\Gamma \vdash C = B_2 \multimap B_1 :$ and $\Gamma; \Delta_1' \vdash P = M_1 : C$ and $\Gamma; \Delta_2' \vdash P = M_2 : C$.
Also (using the definition of erasure), $A_1^- = B_1^- = \tau_1$ and $A_2^- = B_2^- = \tau_2$.
By symmetry and transitivity, $\Gamma \vdash A_2 \multimap A_1 = B_2 \multimap B_1 : \mathsf{Type}$.
By injectivity, $\Gamma \vdash A_2 = B_2 : \mathsf{Type}$ and $\Gamma \vdash A_1 = B_1 : \mathsf{Type}$.
By symmetry and transitivity, $\Gamma \vdash A = B : \mathsf{Type}$.

By type conversion, $\Gamma; \Delta_2'' \vdash N_2 : A_2$ and $\Gamma; \Delta_1' \vdash P = M_1 : A_2 \multimap A_1$ and $\Gamma; \Delta_2' \vdash P = M_2 : A_2 \multimap A_1$.
By the i.h. on the second subderivation, $\Gamma; \Delta_1'' \vdash Q = N_1 : A_2$ and $\Gamma; \Delta_2'' \vdash Q = N_2 : A_2$.
By a congruence rule, $\Gamma; \Delta_1 \vdash P\hat{\ }Q = M_1\hat{\ }N_1 : A_1$ and $\Gamma; \Delta_2 \vdash P\hat{\ }Q = M_2\hat{\ }N_2 : A_1$.
By type conversion, $\Gamma; \Delta_i \vdash P\hat{\ }Q = M_i\hat{\ }N_i : A$.
Finally, using erasure preservation, $A^- = A_1^- = B^- = \tau_1$.

Case:

$$\frac{\Sigma \vdash M_1 \longleftrightarrow M_2 : \tau_1 \,\&\, \tau_2}{\Sigma \vdash \pi_i M_1 \longleftrightarrow \pi_i M_2 : \tau_i}$$

By inversion, $\Gamma; \Delta_1 \vdash M_1 : A_1 \,\&\, A_2$ and $\Gamma; \Delta_2 \vdash M_2 : B_1 \,\&\, B_2$ and $\Gamma \vdash A_i = A : \mathsf{Type}$ and $\Gamma \vdash B_i = B : \mathsf{Type}$.
By the i.h., $\Gamma \vdash C = A_1 \,\&\, A_2 : \mathsf{Type}$ and $\Gamma \vdash C = B_1 \,\&\, B_2 : \mathsf{Type}$ and $\Gamma; \Delta_1 \vdash P = M_1 : C$ and $\Gamma; \Delta_2 \vdash P = M_2 : C$ and $(A_1 \,\&\, A_2)^- = (B_1 \,\&\, B_2)^- = C^- = \tau_1 \,\&\, \tau_2$.
Using the definition of erasure, $A_1^- = B_1^- = \tau_1$ and $A_2^- = B_2^- = \tau_2$.
By symmetry and transitivity, $\Gamma \vdash A_1 \,\&\, A_2 = B_1 \,\&\, B_2 : \mathsf{Type}$.
By injectivity, $\Gamma \vdash A_i = B_i : \mathsf{Type}$.
By symmetry and transitivity, $\Gamma \vdash A = B : \mathsf{Type}$.
By transitivity, $\Gamma \vdash A_i = B : \mathsf{Type}$.
By type conversion, $\Gamma; \Delta_1 \vdash P = M_1 : A_1 \,\&\, A_2$ and $\Gamma; \Delta_2 \vdash P = M_2 : A_1 \,\&\, A_2$.
By a congruence rule, $\Gamma; \Delta_1 \vdash \pi_i P = \pi_i M_1 : A_i$ and $\Gamma; \Delta_2 \vdash \pi_i P = \pi_i M_2 : A_i$.
By type conversion, $\Gamma; \Delta_j \vdash \pi_i P = \pi_i M_j : A$.

Case:

$$\frac{A \xrightarrow{\text{wh}} A' \quad \Gamma^- \vdash A' \Longleftrightarrow B : \mathsf{t}^-}{\Gamma^- \vdash A \Longleftrightarrow B : \mathsf{t}^-}$$

Since $K^- = \mathsf{t}^-$, $K = \mathsf{Type}$.
By Lemma 30, $\Gamma \vdash A = A' : \mathsf{Type}$.
By Lemma 10, $\Gamma \vdash A' : \mathsf{Type}$.
By the i.h., $\Gamma \vdash A' = B : \mathsf{Type}$.
By transitivity, $\Gamma \vdash A = B : \mathsf{Type}$.

Case:

$$\frac{B \xrightarrow{\text{wh}} B' \quad \Gamma^- \vdash A \Longleftrightarrow B' : \mathsf{t}^-}{\Gamma^- \vdash A \Longleftrightarrow B : \mathsf{t}^-}$$

Similar to the previous case.

Case:

$$\frac{\Gamma^- \vdash A \longleftrightarrow B : \mathsf{t}^-}{\Gamma^- \vdash A \Longleftrightarrow B : \mathsf{t}^-}$$

Since $K^- = \mathsf{t}^-$, $K = \mathsf{Type}$. By the i.h., $\Gamma \vdash A = B : \mathsf{Type}$.

Case:

$$\frac{\Gamma^-, u{:}\tau \vdash A\,u \Longleftrightarrow B\,u : \kappa}{\Gamma^- \vdash A \Longleftrightarrow B : \tau \to \kappa}$$

Since $K^- = \tau \to \kappa$, $K = \Pi u{:}C.K'$ where $C^- = \tau$ and $K'^- = \kappa$.
By Lemma 10, $\Gamma \vdash \Pi u{:}C.K' : \mathsf{kind}$.
By Lemma 13, $\Gamma \vdash C : \mathsf{Type}$.
Thus by weakening and rules, $\vdash \Gamma, u{:}C$ $\mathsf{context}$ and $\Gamma, u{:}C \vdash A\,u : K'$ and $\Gamma, u{:}C \vdash B\,u : K'$.
Also, note that $(\Gamma, u{:}C)^- = (\Gamma^-, u{:}\tau)$.
By the i.h., $\Gamma, u{:}C \vdash A\,u = B\,u : K'$.
Thus by extensionality, $\Gamma \vdash A = B : K$.

Case:

$$\frac{}{\Gamma^- \vdash a \longleftrightarrow a : K'^-}\ (S(a) = K')$$

By Lemma 13, $\Gamma \vdash K' = K : \mathsf{kind}$ and $\Gamma \vdash K' = L : \mathsf{kind}$.
By symmetry and transitivity, $\Gamma \vdash K = L : \mathsf{kind}$.
By rule, $\Gamma \vdash a = a : K'$.
By kind conversion, $\Gamma \vdash a = a : K$.
Using erasure preservation, $K^- = L^- = K'^- = \kappa$.

Case:

$$\frac{}{\Gamma^- \vdash \top \longleftrightarrow \top : \mathsf{t}^-}$$

By Lemma 13, $\Gamma \vdash \mathsf{Type} = K : \mathsf{kind}$ and $\Gamma \vdash \mathsf{Type} = L : \mathsf{kind}$.
By symmetry and transitivity, $\Gamma \vdash K = L : \mathsf{kind}$.
By rule, $\Gamma \vdash \top = \top : \mathsf{Type}$.
By kind conversion, $\Gamma \vdash \top = \top : K$.
Using erasure preservation, $K^- = L^- = \mathsf{t}^-$.

Case:

$$\frac{\Gamma^- \vdash A_1 \longleftrightarrow A_2 : \tau \to \kappa \quad \Gamma^- \vdash M_1 \Longleftrightarrow M_2 : \tau}{\Gamma^- \vdash A_1\, M_1 \longleftrightarrow A_2\, M_2 : \kappa}$$

Using Lemma 13, $\Gamma \vdash A_1 : \Pi u{:}B_1.K_1$ and $\Gamma \vdash A_2 : \Pi u{:}B_2.K_2$ and $\Gamma;\epsilon \vdash M_1 : B_1$ and $\Gamma;\epsilon \vdash M_2 : B_2$ and $\Gamma \vdash K_1[M_1/u] = K : \mathsf{kind}$ and $\Gamma \vdash K_2[M_2/u] = L : \mathsf{kind}$.
By the i.h. on the first subderivation, $\Gamma \vdash A_1 = A_2 : \Pi u{:}B_1.K_1$ and $\Gamma \vdash \Pi u{:}B_1.K_1 = \Pi u{:}B_2.K_2 : \mathsf{kind}$ and $(\Pi u{:}B_1.K_1)^- = (\Pi u{:}B_2.K_2)^- = \tau \to \kappa$.
By injectivity, $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$ and $\Gamma, u{:}B_1 \vdash K_1 = K_2 : \mathsf{kind}$.
By symmetry and type conversion, $\Gamma \vdash A_2 : \Pi u{:}B_1.K_1$ and $\Gamma \vdash M_2 : B_1$.
By definition of erasure, $B_1^- = \tau$ and $K_1^- = \kappa$.
By the i.h. on the second subderivation, $\Gamma;\epsilon \vdash P = M_1 : B_1$ and $\Gamma;\epsilon \vdash P = M_2 : B_1$.
By symmetry and transitivity, $\Gamma;\epsilon \vdash M_1 = M_2 : B_1$.
By a congruence rule, $\Gamma \vdash A_1\, M_1 = A_2\, M_2 : K_1[M_1/u]$.
By type conversion, $\Gamma \vdash A_1\, M_1 = A_2\, M_2 : K$.
By kind functionality, $\Gamma \vdash K_1[M_1/u] = K_2[M_2/u] : \mathsf{kind}$.
By erasure preservation, $K^- = K_1^- = K_2^- = L^- = \tau_1$.

Case:

$$\frac{\Gamma^- \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Gamma^-, u{:}A_1^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Gamma^- \vdash \Pi u{:}A_1.B_1 \longleftrightarrow \Pi u{:}A_2.B_2 : \mathsf{t}^-}$$

By Lemma 13, $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ and $\Gamma \vdash L = \mathsf{Type} : \mathsf{kind}$ and $\Gamma \vdash A_1 : \mathsf{Type}$ and $\Gamma \vdash A_2 : \mathsf{Type}$ and $\Gamma, u{:}A_1 \vdash B_1 : \mathsf{Type}$ and $\Gamma, u{:}A_2 \vdash B_2 : \mathsf{Type}$.
By the i.h. on the first subderivation, $\Gamma \vdash A_1 = A_2 : \mathsf{Type}$.
By symmetry and context conversion, $\Gamma, u{:}A_1 \vdash B_2 : \mathsf{Type}$.
Note that $\vdash \Gamma, u{:}A_1$ $\mathsf{context}$ and $(\Gamma, u{:}A_1)^- = (\Gamma^-, u{:}A_1^-)$.
By the i.h. on the second subderivation, $\Gamma, u{:}A_1 \vdash B_1 = B_2 : \mathsf{Type}$.
By a congruence rule, $\Gamma \vdash \Pi u{:}A_1.B_1 = \Pi u{:}A_2.B_2 : \mathsf{Type}$. Using erasure preservation, $K^- = L^- = \mathsf{t}^-$.

Case:

$$\frac{\Gamma^- \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Gamma^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Gamma^- \vdash A_1 \multimap B_1 \longleftrightarrow A_2 \multimap B_2 : \mathsf{t}^-}$$

By Lemma 13, $\Gamma \vdash K = \mathsf{Type} : \mathsf{kind}$ and $\Gamma \vdash L = \mathsf{Type} : \mathsf{kind}$ and $\Gamma \vdash A_1 : \mathsf{Type}$ and $\Gamma \vdash A_2 : \mathsf{Type}$
and $\Gamma \vdash B_1 : \mathsf{Type}$ and $\Gamma \vdash B_2 : \mathsf{Type}$.
By symmetry and transitivity, $\Gamma \vdash K = L : \mathsf{kind}$.
Using equality preservation, $K^- = L^- = \mathsf{t}^-$.

By the i.h., $\Gamma \vdash A_1 = A_2 : \mathsf{Type}$ and $\Gamma \vdash B_1 = B_2 : \mathsf{Type}$.
By a congruence rule, $\Gamma \vdash A_1 \multimap B_1 = A_2 \multimap B_2 : \mathsf{Type}$.

Case:

$$\frac{\Gamma^- \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Gamma^- \vdash B_1 \Longleftrightarrow B_2 : \mathsf{t}^-}{\Gamma^- \vdash A_1 \,\&\, B_1 \longleftrightarrow A_2 \,\&\, B_2 : \mathsf{t}^-}$$

Similar to the previous case.

Case:

$$\frac{}{\Gamma^- \vdash \mathsf{Type} \longleftrightarrow \mathsf{Type} : \mathsf{kind}^-}$$

By rule, $\Gamma \vdash \mathsf{Type} = \mathsf{Type} : \mathsf{kind}$.

Case:

$$\frac{\Gamma^- \vdash A_1 \Longleftrightarrow A_2 : \mathsf{t}^- \quad \Gamma^-, u{:}A_1^- \vdash K_1 \longleftrightarrow K_2 : \mathsf{kind}^-}{\Gamma^- \vdash \Pi u{:}A_1.K_1 \longleftrightarrow \Pi u{:}A_2.K_2 : \mathsf{kind}^-}$$

By Lemma 13, $\Gamma \vdash A_1 : \mathsf{Type}$ and $\Gamma \vdash A_2 : \mathsf{Type}$ and $\Gamma, u{:}A_1 \vdash K_1 : \mathsf{kind}$ and $\Gamma, u{:}A_2 \vdash K_2 : \mathsf{kind}$.
By the i.h. on the first subderivation, $\Gamma \vdash A_1 = A_2 : \mathsf{Type}$.
By symmetry and context conversion, $\Gamma, u{:}A_1 \vdash K_2 : \mathsf{kind}$.
Note that $\vdash \Gamma, u{:}A_1$ context.
By the i.h. on the second subderivation, $\Gamma, u{:}A_1 \vdash K_1 = K_2 : \mathsf{kind}$.
By a congruence rule, $\Gamma \vdash \Pi u{:}A_1.K_1 = \Pi u{:}A_2.K_2 : \mathsf{kind}$.

■

Theorem 2 (Soundness of Algorithmic Equality) *Assume* $\vdash \Gamma$ context *and* $\Gamma \vdash \Delta$ context.

1. *If* $\Gamma; \Delta \vdash M : A$ *and* $\Gamma; \Delta \vdash N : A$ *and* $\Gamma^-, \Delta^- \vdash M \Longleftrightarrow N : A^-$, *then* $\Gamma; \Delta \vdash M = N : A$.
2. *If* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : K$ *and* $\Gamma^- \vdash A \Longleftrightarrow B : K^-$, *then* $\Gamma \vdash A = B : K$.
3. *If* $\Gamma \vdash K : \mathsf{kind}$ *and* $\Gamma \vdash L : \mathsf{kind}$ *and* $\Gamma^- \vdash K \longleftrightarrow L : \mathsf{kind}^-$, *then* $\Gamma \vdash K = L : \mathsf{kind}$.

Proof: Direct, by Lemma 31, symmetry and transitivity.

■

7 Completeness of Algorithmic Equality

In this section we prove that algorithmic equality is complete — that is, that any two terms that are definitionally equal will also be algorithmically equal. To do this, we will define a Kripke logical relation in the style of HP such that logically related terms are algorithmically equal. We will then prove that definitional equality implies the logical relation, thereby establishing completeness.

7.1 A Kripke Logical Relation

Our Kripke logical relation is defined inductively over the same simple types and kinds as were used in the algorithm, and is extended to include substitutions, where it is defined inductively over simple contexts. The worlds are simple contexts, ordered by inclusion. More formally, we will say that a context Σ' extends Σ, written $\Sigma' \succeq \Sigma$, if Σ' contains all the declarations in Σ and possibly more. Our logical relations on terms, families and substitutions are defined as follows:

1. $\Sigma \vdash M = N \in [\![\alpha]\!]$ iff $\Sigma \vdash M \Longleftrightarrow N : \alpha$.
2. $\Sigma \vdash M = N \in [\![\tau_1 \to \tau_2]\!]$ iff for every $\Sigma' \succeq \Sigma$ and every M_1, N_1 such that $\Sigma' \vdash M_1 = N_1 \in [\![\tau_1]\!]$ we have $\Sigma' \vdash M\, M_1 = N\, N_1 \in [\![\tau_2]\!]$.

3. $\Sigma \vdash M = N \in [\![\tau_1 \multimap \tau_2]\!]$ iff for every $\Sigma' \succeq \Sigma$ and every M_1, N_1 such that $\Sigma' \vdash M_1 = N_1 \in [\![\tau_1]\!]$ we have $\Sigma' \vdash M\hat{\ }M_1 = N\hat{\ }N_1 \in [\![\tau_2]\!]$.

4. $\Sigma \vdash M = N \in [\![\tau_1 \,\&\, \tau_2]\!]$ iff $\Sigma \vdash \pi_1 M = \pi_1 N \in [\![\tau_1]\!]$ and $\Sigma \vdash \pi_2 M = \pi_2 N \in [\![\tau_2]\!]$.

5. $\Sigma \vdash M = N \in [\![\top]\!]$, always.

6. $\Sigma \vdash A = B \in [\![\mathsf{t}^-]\!]$ iff $\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-$.

7. $\Sigma \vdash A = B \in [\![\tau \to \kappa]\!]$ iff for every $\Sigma' \succeq \Sigma$ and every M, N such that $\Sigma' \vdash M = N \in [\![\tau]\!]$ we have $\Sigma' \vdash A\,M = B\,N \in [\![\kappa]\!]$.

8. $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\epsilon]\!]$ iff $\sigma_1 = \sigma_2 = \cdot$, the empty substitution.

9. $\Sigma' \vdash \sigma_1[u \mapsto M_1] = \sigma_2[u \mapsto M_2] \in [\![(\Sigma, u{:}\tau)]\!]$ iff $\Sigma' \vdash \sigma_1 = \sigma_2 \in [\![\Sigma]\!]$ and $\Sigma' \vdash M_1 = M_2 \in [\![\tau]\!]$.

Lemma 32 (Monotonicity of Logical Relations) *Let R be any logical relation. If $(\Sigma, \Sigma') \vdash R$ then $(\Sigma, u{:}\tau, \Sigma') \vdash R$.*

Proof: By induction on the type or kind, using the weakening property of algorithmic equality.

■

7.2 Logically Related Terms are Algorithmically Equal

In stating this theorem, we follow HP and prove simultaneously that structurally equal terms are logically related.

Lemma 33 (Logically Related Terms are Algorithmically Equal)

1. *If $\Sigma \vdash M = N \in [\![\tau]\!]$ then $\Sigma \vdash M \Longleftrightarrow N : \tau$.*

2. *If $\Sigma \vdash A = B \in [\![\kappa]\!]$ then $\Sigma \vdash A \Longleftrightarrow B : \kappa$.*

3. *If $\Sigma \vdash M \longleftrightarrow N : \tau$ then $\Sigma \vdash M = N \in [\![\tau]\!]$.*

4. *If $\Sigma \vdash A \longleftrightarrow B : \kappa$ then $\Sigma \vdash A = B \in [\![\kappa]\!]$.*

Proof: By simultaneous structural induction on types and kinds.

Case: Part (1), $\tau = \alpha$. By definition of $[\![\alpha]\!]$, $\Sigma \vdash M \Longleftrightarrow N : \alpha$.

Case: Part (2), $\kappa = \mathsf{t}^-$. By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-$.

Case: Part (3), $\tau = \alpha$.
By a rule, $\Sigma \vdash M \Longleftrightarrow N : \alpha$.
By definition, $\Sigma \vdash M = N \in [\![\alpha]\!]$.

Case: Part (4), $\kappa = \mathsf{t}^-$.
By a rule, $\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-$.
By definition, $\Sigma \vdash A = B \in [\![\mathsf{t}^-]\!]$.

Case: Part (1), $\tau = \tau_1 \to \tau_2$.
By the structural equality rule for variables, $\Sigma, u{:}\tau_1 \vdash u \longleftrightarrow u : \tau_1$.
By the i.h. on τ_1, $\Sigma, u{:}\tau_1 \vdash u = u \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \to \tau_2]\!]$, $\Sigma, u{:}\tau_1 \vdash M\,u = N\,u \in [\![\tau_2]\!]$.
By the i.h. on τ_2, $\Sigma, u{:}\tau_1 \vdash M\,u \Longleftrightarrow N\,u : \tau_2$.
By the type-directed equality rule for $\to$, $\Sigma \vdash M \Longleftrightarrow N : \tau_1 \to \tau_2$.

Case: Part (1), $\tau = \tau_1 \multimap \tau_2$.
By the structural equality rule for variables, $\Sigma, u{:}\tau_1 \vdash u \longleftrightarrow u : \tau_1$.
By the i.h. on τ_1, $\Sigma, u{:}\tau_1 \vdash u = u \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \multimap \tau_2]\!]$, $\Sigma, u{:}\tau_1 \vdash M\hat{\ }u = N\hat{\ }u \in [\![\tau_2]\!]$.
By the i.h. on τ_2, $\Sigma, u{:}\tau_1 \vdash M\hat{\ }u \Longleftrightarrow N\hat{\ }u : \tau_2$.
By the type-directed equality rule for $\multimap$, $\Sigma \vdash M \Longleftrightarrow N : \tau_1 \multimap \tau_2$.

Case: Part (1), $\tau = \tau_1 \,\&\, \tau_2$.
By definition of $[\![\tau_1 \,\&\, \tau_2]\!]$, $\Sigma \vdash \pi_1 M = \pi_1 N \in [\![\tau_1]\!]$ and $\Sigma \vdash \pi_2 M = \pi_2 N \in [\![\tau_2]\!]$.
By the i.h., $\Sigma \vdash \pi_1 M \Longleftrightarrow \pi_1 N : \tau_1$ and $\Sigma \vdash \pi_2 M \Longleftrightarrow \pi_2 N : \tau_2$.
By the type-directed equality rule for &, $\Sigma \vdash M \Longleftrightarrow N : \tau_1 \,\&\, \tau_2$.

Case: Part (1), $\tau = \top$.
By the type-directed equality rule for $\top$, $\Sigma \vdash M \Longleftrightarrow N : \top$.

Case: Part (2), $\kappa = \tau_1 \to \kappa_2$.
By the structural equality rule for variables, $\Sigma, u{:}\tau_1 \vdash u \longleftrightarrow u : \tau_1$.
By the i.h. on τ_1, $\Sigma, u{:}\tau_1 \vdash u = u \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \to \kappa_2]\!]$, $\Sigma, u{:}\tau_1 \vdash M\,u = N\,u \in [\![\kappa_2]\!]$.
By the i.h. on κ_2, $\Sigma, u{:}\tau_1 \vdash M\,u \Longleftrightarrow N\,u : \kappa_2$.
By the kind-directed equality rule for arrow kinds, $\Sigma \vdash M \Longleftrightarrow N : \tau_1 \to \kappa_2$.

Case: Part (3), $\tau = \tau_1 \to \tau_2$.
Suppose that $\Sigma' \succeq \Sigma$ and that $\Sigma' \vdash M' = N' \in [\![\tau_1]\!]$.
By the i.h. on τ_1, $\Sigma' \vdash M' \Longleftrightarrow N' : \tau_1$.
By Lemma 25, $\Sigma' \vdash M \longleftrightarrow N : \tau_1 \to \tau_2$.
By the structural equality rule for application, $\Sigma' \vdash M\,M' \longleftrightarrow N\,N' : \tau_2$.
By the i.h. on τ_2, $\Sigma' \vdash M\,M' = N\,N' \in [\![\tau_2]\!]$.
Thus by definition of $[\![\tau_1 \to \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \to \tau_2]\!]$.

Case: Part (3), $\tau = \tau_1 \multimap \tau_2$.
Suppose that $\Sigma' \succeq \Sigma$ and that $\Sigma' \vdash M' = N' \in [\![\tau_1]\!]$.
By the i.h. on τ_1, $\Sigma' \vdash M' \Longleftrightarrow N' : \tau_1$.
By Lemma 25, $\Sigma' \vdash M \longleftrightarrow N : \tau_1 \multimap \tau_2$.
By the structural equality rule for application, $\Sigma' \vdash M\hat{\ }M' \longleftrightarrow N\hat{\ }N' : \tau_2$.
By the i.h. on τ_2, $\Sigma' \vdash M\hat{\ }M' = N\hat{\ }N' \in [\![\tau_2]\!]$.
Thus by definition of $[\![\tau_1 \multimap \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \multimap \tau_2]\!]$.

Case: Part (3), $\tau = \tau_1 \,\&\, \tau_2$.
By the structural equality rule for projections, $\Sigma \vdash \pi_1\,M \longleftrightarrow \pi_1\,N : \tau_1$ and $\Sigma \vdash \pi_2\,M \longleftrightarrow \pi_2\,N : \tau_2$.
By the i.h., $\Sigma \vdash \pi_1\,M = \pi_1\,N \in [\![\tau_1]\!]$ and $\Sigma \vdash \pi_2\,M = \pi_2\,N \in [\![\tau_2]\!]$.
By definition of $[\![\tau_1 \,\&\, \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \,\&\, \tau_2]\!]$.

Case: Part (4), $\kappa = \tau_1 \to \kappa_2$.
Suppose that $\Sigma' \succeq \Sigma$ and that $\Sigma' \vdash M = N \in [\![\tau_1]\!]$.
By the i.h. on τ_1, $\Sigma' \vdash M \Longleftrightarrow N : \tau_1$.
By Lemma 25, $\Sigma' \vdash A \longleftrightarrow B : \tau_1 \to \kappa_2$.
By the structural equality rule for family application, $\Sigma' \vdash A\,M \longleftrightarrow B\,N : \kappa_2$.
By the i.h. on κ_2, $\Sigma' \vdash A\,M = B\,N \in [\![\kappa_2]\!]$.
Thus by definition of $[\![\tau_1 \to \kappa_2]\!]$, $\Sigma \vdash A = B \in [\![\tau_1 \to \kappa_2]\!]$.

■

7.3 Definitionally Equal Terms are Logically Related

It takes a little more work to prove the next part of completeness, namely that any two terms that are definitionally equal will be related by our Kripke logical relation. As with the logical relation used to prove injectivity, we must prove symmetry, transitivity and closure under head expansion before tackling the proof by induction on definitional equality derivations.

Lemma 34 (Closure under Head Expansion)

1. *If* $M \xrightarrow{\text{wh}} M'$ *and* $\Sigma \vdash M' = N \in [\![\tau]\!]$*, then* $\Sigma \vdash M = N \in [\![\tau]\!]$.
2. *If* $N \xrightarrow{\text{wh}} N'$ *and* $\Sigma \vdash M = N' \in [\![\tau]\!]$*, then* $\Sigma \vdash M = N \in [\![\tau]\!]$.
3. *If* $A \xrightarrow{\text{wh}} A'$ *and* $\Sigma \vdash A' = B \in [\![\kappa]\!]$*, then* $\Sigma \vdash A = B \in [\![\kappa]\!]$.
4. *If* $B \xrightarrow{\text{wh}} B'$ *and* $\Sigma \vdash A = B' \in [\![\kappa]\!]$*, then* $\Sigma \vdash A = B \in [\![\kappa]\!]$.

Proof: Each part is proved by structural induction on the type or kind. We will only show parts (1) and (3); parts (2) and (4) are similar.

Case: Part 1, $\tau = \alpha$.
By definition of $[\![\alpha]\!]$, $\Sigma \vdash M' \Longleftrightarrow N : \tau$.
By a type-directed equality rule, $\Sigma \vdash M \Longleftrightarrow N : \tau$.
By definition of $[\![\alpha]\!]$, $\Sigma \vdash M = N \in [\![\tau]\!]$.

Case: Part 3, $\kappa = \mathsf{t}^-$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash A' \Longleftrightarrow B : \mathsf{t}^-$.
By a kind-directed equality rule, $\Sigma \vdash A \Longleftrightarrow B : \mathsf{t}^-$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash A = B \in [\![\mathsf{t}^-]\!]$.

Case: Part 1, $\tau = \tau_1 \to \tau_2$.
Suppose that $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash M_1 = N_1 \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \to \tau_2]\!]$, $\Sigma' \vdash M'\,M_1 = N\,N_1 \in [\![\tau_2]\!]$,
By a weak head reduction rule, $M\,M_1 \xrightarrow{\text{wh}} M'\,M_1$.
By the i.h. on τ_2, $\Sigma' \vdash M\,M_1 = N\,N_1 \in [\![\tau_2]\!]$,
By definition of $[\![\tau_1 \to \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \to \tau_2]\!]$.

Case: Part 1, $\tau = \tau_1 \multimap \tau_2$.
Suppose that $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash M_1 = N_1 \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \multimap \tau_2]\!]$, $\Sigma' \vdash M'\,\hat{}\,M_1 = N\,\hat{}\,N_1 \in [\![\tau_2]\!]$,
By a weak head reduction rule, $M\,\hat{}\,M_1 \xrightarrow{\text{wh}} M'\,\hat{}\,M_1$.
By the i.h. on τ_2, $\Sigma' \vdash M\,\hat{}\,M_1 = N\,\hat{}\,N_1 \in [\![\tau_2]\!]$,
By definition of $[\![\tau_1 \multimap \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \multimap \tau_2]\!]$.

Case: Part 1, $\tau = \tau_1 \,\&\, \tau_2$.
By definition of $[\![\tau_1 \,\&\, \tau_2]\!]$, $\Sigma \vdash \pi_i M' = \pi_i N \in [\![\tau_i]\!]$ for each i.
By weak head reduction rules, $\pi_i M \xrightarrow{\text{wh}} \pi_i M'$ for each i.
By the i.h. on each τ_i, $\Sigma \vdash \pi_i M = \pi_i N \in [\![\tau_i]\!]$.
By definition of $[\![\tau_1 \,\&\, \tau_2]\!]$, $\Sigma \vdash M = N \in [\![\tau_1 \,\&\, \tau_2]\!]$.

Case: Part 1, $\tau = \top$.
By definition of $[\![\top]\!]$, trivially $\Sigma \vdash M = N \in [\![\top]\!]$.

Case: Part 3, $\kappa = \tau_1 \to \kappa_2$.
Suppose that $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash M = N \in [\![\tau_1]\!]$.
By definition of $[\![\tau_1 \to \kappa_2]\!]$, $\Sigma' \vdash A'\,M = B\,N \in [\![\kappa_2]\!]$.

By a weak head reduction rule, $A\,M \xrightarrow{\text{wh}} A'\,M$.
By the i.h. on κ_2, $\Sigma' \vdash A\,M = B\,N \in [\![\kappa_2]\!]$.
By definition of $[\![\tau_1 \to \kappa_2]\!]$, $\Sigma \vdash A = B \in [\![\tau_1 \to \kappa_2]\!]$.

■

Lemma 35 (Symmetry of the Logical Relations)

1. *If* $\Sigma \vdash M = N \in [\![\tau]\!]$*, then* $\Sigma \vdash N = M \in [\![\tau]\!]$.
2. *If* $\Sigma \vdash A = B \in [\![\kappa]\!]$*, then* $\Sigma \vdash B = A \in [\![\kappa]\!]$.

Proof: Straightforward, by induction on types and kinds and using Lemma 28.

■

Lemma 36 (Transitivity of the Logical Relations)

1. *If* $\Sigma \vdash M = N \in [\![\tau]\!]$ *and* $\Sigma \vdash N = O \in [\![\tau]\!]$ *then* $\Sigma \vdash M = O \in [\![\tau]\!]$.
2. *If* $\Sigma \vdash A = B \in [\![\kappa]\!]$ *and* $\Sigma \vdash B = C \in [\![\kappa]\!]$ *then* $\Sigma \vdash A = C \in [\![\kappa]\!]$.

Proof: By induction on types and kinds, using the previous lemma and Lemma 29.

■

Now we can prove the main lemma of this section, namely that logically related substitutions map definitionally equal terms to logically related terms. Once we have done this we can use the fact that identity substitutions are logically related to establish completeness of the algorithm. Because the definitional equality judgments enforce linearity but the logical relations do not, the statement of the main lemma must allow the domain of the substitutions to contain variables that are not declared in the context of the equality judgment.

Lemma 37 (Definitionally Equal Terms are Logically Related under Substitutions)

1. *If* $\Gamma;\Delta \vdash M_1 = M_2 : A$ *and* $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta^-, \Theta]\!]$ *then* $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^-]\!]$.
2. *If* $\Gamma \vdash A_1 = A_2 : K$ *and* $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Theta]\!]$ *then* $\Sigma \vdash \sigma_1 A_1 = \sigma_2 A_2 \in [\![K^-]\!]$.

Proof: By induction on the derivation of definitional equality.

Case:

$$\frac{}{\Gamma;\epsilon \vdash u = u : A}\ (\Gamma(u) = A)$$

By Lemmas 10 and 13, $\Gamma(u) = B$ where $\Gamma \vdash A = B : \mathsf{Type}$.
By erasure preservation, $A^- = B^-$.
By the definition of erasure, $\Delta^- = \epsilon$ and $(u{:}A^-) \in \Gamma^-$.
Thus by definition of $[\![\Gamma^-, \Delta^-, \Theta]\!]$, $\Sigma \vdash \sigma_1(u) = \sigma_2(u) \in [\![A^-]\!]$.

Case:

$$\frac{}{\Gamma;\epsilon \vdash c = c : A}\ (S(c) = A)$$

Observe that $\sigma_i c = c$ for each i.
By the structural equality rule for constants, $\Sigma \vdash c \longleftrightarrow c : A^-$.
That is, $\Sigma \vdash \sigma_1 c \longleftrightarrow \sigma_2 c : A^-$.
By Lemma 33, $\Sigma \vdash \sigma_1 c = \sigma_2 c \in [\![A^-]\!]$.

Case:

$$\frac{}{\Gamma; u\hat{:}A \vdash u = u : A}$$

By definition of erasure, $\Delta^- = u\hat{:}A^-$.
Thus by definition of $[\![\Gamma^-, \Delta^-, \Theta]\!]$, $\Sigma \vdash \sigma_1(u) = \sigma_2(u) \in [\![A^-]\!]$.

Case:

$$\frac{\Gamma \vdash A_1 = A : \mathsf{Type} \quad \Gamma \vdash A_2 = A : \mathsf{Type} \quad \Gamma; \Delta, u\hat{:}A \vdash M_1 = M_2 : B}{\Gamma; \Delta \vdash \hat{\lambda} u{:}A_1.M_1 = \hat{\lambda} u{:}A_2 M_2 : A \multimap B}$$

WLOG, we may assume $u \notin \mathrm{Dom}(\Theta)$.
Note that $(A \multimap B)^- = A^- \multimap B^-$.
So, suppose $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash N_1 = N_2 \in [\![A^-]\!]$.
By Lemma 32, $\Sigma' \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta^-, \Theta]\!]$.
By definition of the logical relation, $\Sigma' \vdash \sigma_1[u \mapsto N_1] = \sigma_2[u \mapsto N_2] \in [\![\Gamma^-, \Delta^-, u{:}A^-, \Theta]\!]$.
By the i.h., $\Sigma' \vdash \sigma_1[u \mapsto N_1]M_1 = \sigma_2[u \mapsto N_2]M_2 \in [\![B^-]\!]$.
That is, $\Sigma' \vdash (\sigma_1 M_1)[N_1/u] = (\sigma_2 M_2)[N_2/u] \in [\![B^-]\!]$.
By Lemma 34, $\Sigma' \vdash (\hat{\lambda} u{:}\sigma_1 A_1.\sigma_1 M_1)\,\hat{}\,N_1 = (\sigma_2 M_2)[N_2/u] \in [\![B^-]\!]$.
By Lemma 34, $\Sigma' \vdash (\hat{\lambda} u{:}\sigma_1 A_1.\sigma_1 M_1)\,\hat{}\,N_1 = (\hat{\lambda} u{:}\sigma_2 A_2.\sigma_2 M_2)\,\hat{}\,N_2 \in [\![B^-]\!]$.
Thus by definition of $[\![A^- \multimap B^-]\!]$, $\Sigma \vdash \hat{\lambda} u{:}\sigma_1 A_1.\sigma_1 M_1 = \hat{\lambda} u{:}\sigma_2 A_2.\sigma_2 M_2 \in [\![A^- \multimap B^-]\!]$.
That is, $\Sigma \vdash \sigma_1(\hat{\lambda} u{:}A_1.M_1) = \sigma_2(\hat{\lambda} u{:}A_2.M_2) \in [\![A^- \multimap B^-]\!]$.

Case:

$$\frac{\cdots \quad \Gamma, u{:}A; \Delta \vdash M_1 = M_2 : B}{\Gamma; \Delta \vdash \lambda u{:}A_1.M_1 = \lambda u{:}A_2 M_2 : \Pi u{:}A.B}$$

Similar to the previous case.

Case:

$$\frac{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \quad \Gamma; \Delta \vdash N_1 = N_2 : A_2}{\Gamma; \Delta \vdash \langle M_1, N_1 \rangle = \langle M_2, N_2 \rangle : A_1 \mathbin{\&} A_2}$$

By the i.h., $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A_1^-]\!]$ and $\Sigma \vdash \sigma_1 N_1 = \sigma_2 N_2 \in [\![A_2^-]\!]$.
Using Lemma 34, $\Sigma \vdash \pi_1 \langle \sigma_1 M_1, \sigma_1 N_1 \rangle = \pi_1 \langle \sigma_2 M_2, \sigma_2 N_2 \rangle \in [\![A_1^-]\!]$
and $\Sigma \vdash \pi_2 \langle \sigma_1 M_1, \sigma_1 N_1 \rangle = \pi_2 \langle \sigma_2 M_2, \sigma_2 N_2 \rangle \in [\![A_2^-]\!]$.
By definition of $[\![A_1^- \mathbin{\&} A_2^-]\!]$, $\Sigma \vdash \langle \sigma_1 M_1, \sigma_1 N_1 \rangle = \langle \sigma_2 M_2, \sigma_2 N_2 \rangle \in [\![A_1^- \mathbin{\&} A_2^-]\!]$.
That is, $\Sigma \vdash \sigma_1 \langle M_1, N_1 \rangle = \sigma_2 \langle M_2, N_2 \rangle \in [\![A_1^- \mathbin{\&} A_2^-]\!]$.

Case:

$$\frac{\Gamma; \Delta_1 \vdash M_1 = M_2 : A \multimap B \quad \Gamma; \Delta_2 \vdash N_1 = N_2 : A}{\Gamma; \Delta_1, \Delta_2 \vdash M_1\,\hat{}\,N_1 = M_2\,\hat{}\,N_2 : B}$$

First, note that $(A \multimap B)^- = A^- \multimap B^-$.
Let $\Theta_1 = \Delta_2^-, \Theta$. Then $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta_1^-, \Theta_1]\!]$.
By the i.h. on the first subderivation, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^- \multimap B^-]\!]$.
Let $\Theta_2 = \Delta_1^-, \Theta$. Then $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta_2^-, \Theta_2]\!]$.
By the i.h. on the second subderivation, $\Sigma \vdash \sigma_1 N_1 = \sigma_2 N_2 \in [\![A^-]\!]$.
By definition of $[\![A^- \multimap B^-]\!]$, $\Sigma \vdash (\sigma_1 M_1)\,\hat{}\,(\sigma_1 N_1) = (\sigma_2 M_2)\,\hat{}\,(\sigma_2 N_2) \in [\![B^-]\!]$.
That is, $\Sigma \vdash \sigma_1(M_1\,\hat{}\,N_1) = \sigma_2(M_2\,\hat{}\,N_2) \in [\![B^-]\!]$.

Case:

$$\frac{\Gamma; \Delta \vdash M_1 = N_1 : \Pi u{:}A.B \quad \Gamma; \epsilon \vdash N_1 = N_2 : A}{\Gamma; \Delta \vdash M_1\, N_1 = M_2\, N_2 : B[N_1/x]}$$

Similar to the previous case.

Case:

$$\frac{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \mathbin{\&} A_2}{\Gamma; \Delta \vdash \pi_i M_1 = \pi_i M_2 : A_i}$$

By the i.h., $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A_1^- \mathbin{\&} A_2^-]\!]$.
By definition of $[\![A_1^- \mathbin{\&} A_2^-]\!]$, $\Sigma \vdash \pi_i(\sigma_1 M_1) = \pi_i(\sigma_2 M_2) \in [\![A_i^-]\!]$.
That is, $\Sigma \vdash \sigma_1(\pi_i M_1) = \sigma_1(\pi_i M_2) \in [\![A_i^-]\!]$.

Case:

$$\frac{\Gamma;\Delta \vdash M_1 = M_2 : A' \quad \Gamma \vdash A' = A : \mathsf{Type}}{\Gamma;\Delta \vdash M_1 = M_2 : A}$$

By the i.h., $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A'^-]\!]$.
By Lemma 24, $A^- = A'^-$.
Thus, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^-]\!]$.

Case:

$$\frac{\Gamma;\Delta \vdash M_2 = M_1 : A}{\Gamma;\Delta \vdash M_1 = M_2 : A}$$

By Lemma 35, $\Sigma \vdash \sigma_2 = \sigma_1 \in [\![\Gamma^-, \Delta^-, \Theta]\!]$.
By the i.h., $\Sigma \vdash \sigma_2 M_2 = \sigma_1 M_1 \in [\![A^-]\!]$.
By Lemma 35, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^-]\!]$.

Case:

$$\frac{\Gamma;\Delta \vdash M_1 = M_3 : A \quad \Gamma;\Delta \vdash M_3 = M_2 : A}{\Gamma;\Delta \vdash M_1 = M_2 : A}$$

By Lemmas 35 and 36, $\Sigma \vdash \sigma_1 = \sigma_1 \in [\![\Gamma^-, \Delta^-, \Theta]\!]$.
By the i.h. on the first subderivation, $\Sigma \vdash \sigma_1 M_1 = \sigma_1 M_3 \in [\![A^-]\!]$.
By the i.h. on the second subderivation, $\Sigma \vdash \sigma_1 M_3 = \sigma_2 M_2 \in [\![A^-]\!]$.
By Lemma 36, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^-]\!]$.

Case:

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma;\Delta_1, u\hat{:}A \vdash M_1 = M_2 : B \quad \Gamma;\Delta_2 \vdash N_1 = N_2 : A}{\Gamma;\Delta_1,\Delta_2 \vdash (\hat{\lambda} u{:}A.M_1)\,\hat{}\,N_1 = M_2[N_2/u] : B}$$

Let $\Theta_1 = \Delta_2^-, \Theta$ and $\Theta_2 = \Delta_1^-, \Theta$.
Then we know $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta_1^-, \Theta_1]\!]$ and $\Sigma \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta_2^-, \Theta_2]\!]$.
By the induction hypothesis on the second equality subderivation, $\Sigma \vdash \sigma_1 N_1 = \sigma_2 N_2 \in [\![A^-]\!]$.
By definition of the logical relation, $\Sigma \vdash \sigma_1[u{\mapsto}\sigma_1 N_1] = \sigma_2[u{\mapsto}\sigma_2 N_2] \in [\![\Gamma^-, \Delta_1^-, u{:}A^-, \Delta_2^-, \Theta]\!]$.
By the induction hypothesis on the first equality subderivation,
  $\Sigma \vdash \sigma_1[u{\mapsto}\sigma_1 N_1]M_1 = \sigma_2[u{\mapsto}\sigma_2 N_2]M_2 \in [\![B^-]\!]$.
That is, $\Sigma \vdash \sigma_1 M_1[\sigma_1 N_1/u] = \sigma_2 M_2[\sigma_2 N_2/u] \in [\![B^-]\!]$.
Now, observe that $\sigma_1((\hat{\lambda} u{:}A.M_1)\,\hat{}\,N_1) = (\hat{\lambda} u{:}\sigma_1 A.\sigma_1 M_1)\,\hat{}\,(\sigma_1 N_1) \xrightarrow{\text{wh}} \sigma_1 M_1[\sigma_1 N_1/u]$
and that $\sigma_2(M_2[N_2/u]) = \sigma_2 M_2[\sigma_2 N_2/u]$.
Thus by Lemma 34, $\Sigma \vdash \sigma_1((\lambda u{:}A.M_1)\,N_1) = \sigma_2(M_2[N_2/u]) \in [\![B^-]\!]$.

Case:

$$\frac{\Gamma \vdash A : \mathsf{Type} \quad \Gamma, u{:}A;\Delta \vdash M_1 = M_2 : B \quad \Gamma;\epsilon \vdash N_1 = N_2 : A}{\Gamma;\Delta \vdash (\lambda u{:}A.M_1)\,N_1 = M_2[N_2/u] : B[N_1/u]}$$

Similar to the previous case.

Case:

$$\frac{\Gamma;\Delta \vdash M_1 = N_1 : A_1 \quad \Gamma;\Delta \vdash M_2 = N_2 : A_2}{\Gamma;\Delta \vdash \pi_i\langle M_1, M_2\rangle = N_i : A_i}$$

By the i.h., $\Sigma \vdash \sigma_1 M_i = \sigma_2 N_i \in [\![A_i^-]\!]$ for $i = 1, 2$.
Observe that $\sigma_1(\pi_i\langle M_1, M_2\rangle) = \pi_i\langle \sigma_1 M_1, \sigma_1 M_2\rangle \xrightarrow{\text{wh}} \sigma_1 M_i$.

Thus by Lemma 34, $\Sigma \vdash \sigma_1(\pi_i\langle M_1, M_2\rangle) = \sigma_2 N_i \in [\![A_i^-]\!]$.

Case:

$$\frac{\cdots \quad \Gamma; \Delta, u\hat{:}A \vdash M_1\,\hat{}\,u = M_2\,\hat{}\,u : B}{\Gamma; \Delta \vdash M_1 = M_2 : A \multimap B}$$

WLOG, we may assume u is fresh, *i.e.* it does not appear free in M_1 or M_2 and is not in $\mathrm{Dom}(\Theta)$.
So, suppose $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash N_1 = N_2 \in [\![A^-]\!]$.
By Lemma 32, $\Sigma' \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Delta^-, \Theta]\!]$.
By definition of the logical relation, $\Sigma' \vdash \sigma_1[u \mapsto N_1] = \sigma_2[u \mapsto N_2] \in [\![\Gamma^-, \Delta^-, u{:}A^-, \Theta]\!]$.
By the i.h., $\Sigma' \vdash \sigma_1[u \mapsto N_1](M_1\,\hat{}\,u) = \sigma_2[u \mapsto N_2](M_2\,\hat{}\,u) \in [\![B^-]\!]$.
That is, $\Sigma' \vdash (\sigma_1 M_1)\,\hat{}\,N_1 = (\sigma_2 M_2)\,\hat{}\,N_2 \in [\![B^-]\!]$.
By definition of $[\![A^- \multimap B^-]\!]$, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A^- \multimap B^-]\!]$.

Case:

$$\frac{\cdots \quad \Gamma, u{:}A; \Delta \vdash M_1\, u = M_2\, u : B}{\Gamma; \Delta \vdash M_1 = M_2 : \Pi u{:}A.B}$$

Similar to the previous case.

Case:

$$\frac{\cdots \quad \Gamma; \Delta \vdash \pi_i M_1 = \pi_i M_2 : A_i \text{ for } i = 1, 2}{\Gamma; \Delta \vdash M_1 = M_2 : A_1 \,\&\, A_2}$$

By the i.h., $\Sigma \vdash \sigma_1(\pi_i M_1) = \sigma_2(\pi_i M_2) \in [\![A_i^-]\!]$ for $i = 1, 2$.
That is, $\Sigma \vdash \pi_i(\sigma_1 M_1) = \pi_i(\sigma_2 M_2) \in [\![A_i^-]\!]$.
By definition of $[\![A_1^- \,\&\, A_2^-]\!]$, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![A_1^- \,\&\, A_2^-]\!]$.

Case:

$$\frac{\cdots}{\Gamma; \Delta \vdash M = N : \top}$$

By definition of $[\![\top]\!]$, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![\top]\!]$.

Case:

$$\frac{}{\Gamma \vdash a = a : K}\ (S(a) = K)$$

By the structural family equality rule for constants, $\Sigma \vdash a \longleftrightarrow a : K^-$.
That is, $\Sigma \vdash \sigma_1 a \longleftrightarrow \sigma_2 a : K^-$.
By Lemma 33, $\Sigma \vdash \sigma_1 a = \sigma_2 a \in [\![K^-]\!]$.

Case:

$$\frac{}{\Gamma \vdash \top = \top : \mathsf{Type}}$$

Note that $\sigma_i \top = \top$.
By the structural familiy equality rule for $\top$, $\Sigma \vdash \top \longleftrightarrow \top : \mathsf{t}^-$.
By Lemma 33, $\Sigma \vdash \top = \top \in [\![\mathsf{t}^-]\!]$.

Case:

$$\frac{\cdots \quad \Gamma, u{:}A \vdash B_1 = B_2 : K}{\Gamma \vdash \lambda u{:}A_1.B_1 = \lambda u{:}A_2.B_2 : \Pi u{:}A.K}$$

WLOG, we may assume $u \notin \mathrm{Dom}(\Theta)$.
Note that $(\Pi u{:}A.K)^- = A^- \to K^-$.
So, suppose $\Sigma' \succeq \Sigma$ and $\Sigma' \vdash N_1 = N_2 \in [\![A^-]\!]$.
By Lemma 32, $\Sigma' \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Theta]\!]$.
By definition of the logical relation, $\Sigma' \vdash \sigma_1[u \mapsto N_1] = \sigma_2[u \mapsto N_2] \in [\![\Gamma^-, u{:}A^-, \Theta]\!]$.

By the i.h., $\Sigma' \vdash \sigma_1[u{\mapsto}N_1]B_1 = \sigma_2[u{\mapsto}N_2]B_2 \in [\![K^-]\!]$.
That is, $\Sigma' \vdash (\sigma_1 B_1)[N_1/u] = (\sigma_2 B_2)[N_2/u] \in [\![K^-]\!]$.
By Lemma 34, $\Sigma' \vdash (\lambda u{:}\sigma_1 A_1.\sigma_1 B_1)\, N_1 = (\sigma_2 B_2)[N_2/u] \in [\![K^-]\!]$.
By Lemma 34, $\Sigma' \vdash (\lambda u{:}\sigma_1 A_1.\sigma_1 B_1)\, N_1 = (\lambda u{:}\sigma_2 A_2.\sigma_2 B_2)\, N_2 \in [\![K^-]\!]$.
Thus by definition of $[\![A^- \to K^-]\!]$, $\Sigma \vdash \lambda u{:}\sigma_1 A_1.\sigma_1 B_1 = \lambda u{:}\sigma_2 A_2.\sigma_2 B_2 \in [\![A^- \to K^-]\!]$.
That is, $\Sigma \vdash \sigma_1(\lambda u{:}A_1.B_1) = \sigma_2(\lambda u{:}A_2.B_2) \in [\![A^- \to K^-]\!]$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K \quad \Gamma; \epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash A_1\, M_1 = A_2\, M_2 : K[M_1/u]}$$

By Lemma 23, $(K[M_1/u])^- = K^-$.
By the i.h. on the first subderivation, $\Sigma \vdash \sigma_1 A_1 = \sigma_2 A_2 \in [\![B^- \to K^-]\!]$.
By the i.h. on the second subderivation, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![B^-]\!]$.
By definition of $[\![B^- \to K^-]\!]$, $\Sigma \vdash (\sigma_1 A_1)(\sigma_1 M_1) = (\sigma_2 A_2)(\sigma_2 M_2) \in [\![K^-]\!]$.
That is, $\Sigma \vdash \sigma_1(A_1\, M_1) = \sigma_2(A_2\, M_2) \in [\![K^-]\!]$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma, u{:}A_1 \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash \Pi u{:}A_1.B_1 = \Pi u{:}A_2.B_2 : \mathsf{Type}}$$

WLOG, assume $u \notin \mathrm{Dom}(\Theta)$.
By Lemma 32, $\Sigma, u{:}A_1^- \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Theta]\!]$.
By the structural object equality rule for variables, $\Sigma, u{:}A_1^- \vdash u \longleftrightarrow u : A_1^-$.
By Lemma 33, $\Sigma, u{:}A_1^- \vdash u = u \in [\![A_1^-]\!]$.
By definition of the logical relation, $\Sigma, u{:}A_1^- \vdash \sigma_1[u{\mapsto}u] = \sigma_2[u{\mapsto}u] \in [\![(\Gamma^-, u{:}A_1^-), \Theta]\!]$.
By the i.h. on the second subderivation, $\Sigma, u{:}A_1^- \vdash \sigma_1[u{\mapsto}u]B_1 = \sigma_2[u{\mapsto}u]B_2 \in [\![\mathsf{t}^-]\!]$.
That is, $\Sigma, u{:}A_1^- \vdash \sigma_1 B_1 = \sigma_2 B_2 \in [\![\mathsf{t}^-]\!]$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma, u{:}A_1^- \vdash \sigma_1 B_1 \Longleftrightarrow \sigma_2 B_2 : \mathsf{t}^-$.
By the i.h. on the first subderivation, $\Sigma \vdash \sigma_1 A_1 = \sigma_2 A_2 \in [\![\mathsf{t}^-]\!]$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash \sigma_1 A_1 \Longleftrightarrow \sigma_2 A_2 : \mathsf{t}^-$.
By the structural family equality rule for Π-families, $\Sigma \vdash \sigma_1(\Pi u{:}A_1.B_1) \longleftrightarrow \sigma_2(\Pi u{:}A_2.B_2) : \mathsf{t}^-$.
By the kind-directed family equality rule for t^-, $\Sigma \vdash \sigma_1(\Pi u{:}A_1.B_1) \Longleftrightarrow \sigma_2(\Pi u{:}A_2.B_2) : \mathsf{t}^-$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash \sigma_1(\Pi u{:}A_1.B_1) = \sigma_2(\Pi u{:}A_2.B_2) \in [\![\mathsf{t}^-]\!]$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \multimap B_1 = A_2 \multimap B_2 : \mathsf{Type}}$$

By the i.h. on the first subderivation, $\Sigma \vdash \sigma_1 A_1 = \sigma_2 A_2 \in [\![\mathsf{t}^-]\!]$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash \sigma_1 A_1 \Longleftrightarrow \sigma_2 A_2 : \mathsf{t}^-$.
By the i.h. on the second subderivation, $\Sigma \vdash \sigma_1 B_1 = \sigma_2 B_2 \in [\![\mathsf{t}^-]\!]$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash \sigma_1 B_1 \Longleftrightarrow \sigma_2 B_2 : \mathsf{t}^-$.
By a structural family equality rule, $\Sigma \vdash (\sigma_1 A_1) \multimap (\sigma_1 B_1) \longleftrightarrow (\sigma_2 A_2) \multimap (\sigma_2 B_2) : \mathsf{t}^-$.
That is, $\Sigma \vdash \sigma_1(A_1 \multimap B_1) \longleftrightarrow \sigma_2(A_2 \multimap B_2) : \mathsf{t}^-$.
By the kind-directed family equality rule for t^-, $\Sigma \vdash \sigma_1(A_1 \multimap B_1) \Longleftrightarrow \sigma_2(A_2 \multimap B_2) : \mathsf{t}^-$.
By definition of $[\![\mathsf{t}^-]\!]$, $\Sigma \vdash \sigma_1(A_1 \multimap B_1) = \sigma_2(A_2 \multimap B_2) \in [\![\mathsf{t}^-]\!]$.

Case:

$$\frac{\Gamma \vdash A_1 = A_2 : \mathsf{Type} \quad \Gamma \vdash B_1 = B_2 : \mathsf{Type}}{\Gamma \vdash A_1 \& B_1 = A_2 \& B_2 : \mathsf{Type}}$$

Similar to the previous case.

Case:

$$\frac{\Gamma \vdash A_2 = A_1 : K}{\Gamma \vdash A_1 = A_2 : K} \quad \text{or} \quad \frac{\Gamma \vdash A_1 = A_3 : K \quad \Gamma \vdash A_3 = A_2 : K}{\Gamma \vdash A_1 = A_2 : K} \quad \text{or} \quad \frac{\Gamma \vdash A_1 = A_2 : K' \quad \Gamma \vdash K' = K : \mathsf{kind}}{\Gamma \vdash A_1 = A_2 : K}$$

Similar to the corresponding cases for objects.

Case:

$$\frac{\cdots \quad \Gamma, u{:}B \vdash A_1 = A_2 : K \quad \Gamma; \epsilon \vdash M_1 = M_2 : B}{\Gamma \vdash (\lambda u{:}B.A_1)\, M_1 = A_2[M_2/u] : K[M_1/u]}$$

By the i.h. on the object equality subderivation, $\Sigma \vdash \sigma_1 M_1 = \sigma_2 M_2 \in [\![B^-]\!]$.
By definition of the logical relation, $\Sigma \vdash \sigma_1[u{\mapsto}\sigma_1 M_1] = \sigma_2[u{\mapsto}\sigma_2 M_2] \in [\![\Gamma^-, u{:}B^-, \Theta]\!]$.
By the i.h. on the family equality subderivation, $\Sigma \vdash \sigma_1[u{\mapsto}\sigma_1 M_1]A_1 = \sigma_2[u{\mapsto}\sigma_2 M_2]A_2 \in [\![K^-]\!]$.
That is, $\Sigma \vdash \sigma_1 A_1[\sigma_1 M_1/u] = \sigma_2 A_2[\sigma_2 M_2/u] \in [\![K^-]\!]$.
Now, observe that $\sigma_1((\lambda u{:}B.A_1)\, M_1) = (\lambda u{:}\sigma_1 B.\sigma_1 A_1)\,(\sigma_1 M_1) \xrightarrow{\text{wh}} \sigma_1 A_1[\sigma_1 M_1/u]$
and that $\sigma_2(A_2[M_2/u]) = (\sigma_2 A_2)[\sigma_2 M_2/u]$.
Thus by Lemma 34, $\Sigma \vdash \sigma_1((\lambda u{:}B.A_1)\, M_1) = \sigma_2(A_2[M_2/u]) \in [\![K^-]\!]$.

Case:

$$\frac{\Gamma \vdash B : \mathsf{Type} \quad \Gamma \vdash A_1 : \Pi u{:}B.K \quad \Gamma \vdash A_2 : \Pi u{:}B.K \quad \Gamma, u{:}B \vdash A_1\, u = A_2\, u : K}{\Gamma \vdash A_1 = A_2 : \Pi u{:}B.K}$$

WLOG, we may assume u is fresh.
Note that $(\Pi u{:}B.K)^- = B^- \to K^-$.
So, suppose $\Sigma' \succeq \Sigma$ and $\Sigma \vdash M_1 = M_2 \in [\![B^-]\!]$.
By Lemma 32, $\Sigma' \vdash \sigma_1 = \sigma_2 \in [\![\Gamma^-, \Theta]\!]$.
By definition of the logical relation, $\Sigma' \vdash \sigma_1[u{\mapsto}M_1] = \sigma_2[u{\mapsto}M_2] \in [\![\Gamma^-, u{:}B^-, \Theta]\!]$.
By the i.h., $\Sigma' \vdash \sigma_1[u{\mapsto}M_1](A_1\, u) = \sigma_2[u{\mapsto}M_2](A_2\, u) \in [\![K^-]\!]$.
That is, $\Sigma' \vdash (\sigma_1 A_1)\, M_1 = (\sigma_2 A_2)\, M_2 \in [\![K^-]\!]$.
By definition of $[\![B^- \to K^-]\!]$, $\Sigma \vdash \sigma_1 A_1 = \sigma_2 A_2 \in [\![B^- \to K^-]\!]$.

Lemma 38 (Identity Substitutions are Logically Related) $\Sigma \vdash \mathrm{id}_\Sigma = \mathrm{id}_\Sigma \in [\![\Sigma]\!]$.

Proof: By Lemma 33 and the definition of the logical relation for substitutions.

Theorem 3 (Completeness of Algorithmic Equality)

- *If* $\Gamma; \Delta \vdash M = N : A$ *then* $\Gamma^-, \Delta^- \vdash M \Longleftrightarrow N : A^-$.
- *If* $\Gamma \vdash A = B : K$ *then* $\Gamma^- \vdash A \Longleftrightarrow B : K^-$

Proof: By Lemmas 33, 37 and 38.

8 Decidability of Equality

Having established soundness and completeness for our algorithmic equality judgments, we may prove that equality is decidable — in effect, that the algorithmic rules do in fact define an algorithm — in exactly the same way as HP.

The proof of decidability is split into two parts. First, we prove that algorithmic equality is decidable when each of the terms being compared is algorithmically equal to some other term. Then, we can use this fact to prove that definitional equality is decidable for all well-typed terms by noting that any well-typed term is algorithmically equal to itself. Following HP, we will call a term *normalizing* if it is algorithmically equal to some other term—this terminology reflects the fact that a canonical form for the two terms can be extracted from the algorithmic equality derivation.

Lemma 39 (Decidability for Normalizing Terms)

1. *If* $\Sigma \vdash M \Longleftrightarrow M' : \tau$ *and* $\Sigma \vdash N \Longleftrightarrow N' : \tau$ *then it is decidable whether* $\Sigma \vdash M \Longleftrightarrow N : \tau$.
2. *If* $\Sigma \vdash M \longleftrightarrow M' : \tau_1$ *and* $\Sigma \vdash N \longleftrightarrow N' : \tau_2$ *then it is decidable whether* $\Sigma \vdash M \longleftrightarrow N : \tau_3$ *for some* τ_3.
3. *If* $\Sigma \vdash A \Longleftrightarrow A' : \kappa$ *and* $\Sigma \vdash B \Longleftrightarrow B' : \kappa$ *then it is decidable whether* $\Sigma \vdash A \Longleftrightarrow B : \kappa$.
4. *If* $\Sigma \vdash A \longleftrightarrow A' : \kappa_1$ *and* $\Sigma \vdash B \longleftrightarrow B' : \kappa_2$ *then it is decidable whether* $\Sigma \vdash A \longleftrightarrow B : \kappa_3$ *for some* κ_3.
5. *If* $\Sigma \vdash K \longleftrightarrow K' : \mathsf{kind}^-$ *and* $\Sigma \vdash L \longleftrightarrow L' : \mathsf{kind}^-$ *then it is decidable whether* $\Sigma \vdash K \longleftrightarrow L : \mathsf{kind}^-$.

Proof: By induction on the given derivations, using determinacy (Lemmas 15 and 26).

■

Theorem 4 (Decidability for Well-Formed Terms)

1. *If* $\Gamma; \Delta \vdash M : A$ *and* $\Gamma; \Delta \vdash N : A$ *then it is decidable whether* $\Gamma; \Delta \vdash M = N : A$.
2. *If* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : K$ *then it is decidable whether* $\Gamma \vdash A = B : K$.
3. *If* $\Gamma \vdash K : \mathsf{kind}$ *and* $\Gamma \vdash L : \mathsf{kind}$ *then it is decidable whether* $\Gamma \vdash K = L : \mathsf{kind}$.

Proof: Because algorithmic equality is sound (Theorem 2) and complete (Theorem 3), it suffices to check algorithmic equality in each case. Furthermore, by reflexivity (Lemma 3) and completeness, each term is algorithmically equal to itself. Thus by Lemma 39, algorithmic equality of the two terms is decidable.

■

9 Conclusion

We have presented a variant of the LLF type theory in which terms need not be in pre-canonical form in order to be well-typed. Our variant differs from the original presentation of LLF by Cervesato and Pfenning by employing a set of typed definitional equality judgments rather than taking definitional equality to be untyped β- or $\beta\eta$-conversion. We have proved that this notion of definitional equality for well-typed terms is decidable by giving a type-directed algorithm and proving it sound and complete. The equality algorithm is simplified by the identification of intuitionistic and linear assumptions, relying on the well-formedness of the terms being compared to ensure linearity is respected.

We have not addressed the problem of finding canonical forms for terms, or proving that they exist. However, we believe that our algorithm, like that of Harper and Pfenning on which it is based, can be instrumented to extract canonical forms for the terms it compares. In fact, a trick very similar to this instrumentation is performed implicitly in our soundness proof, where algorithmically equal terms are proved definitionally equal by extracting a mediating term. It appears that this mediating term is canonical except for the type labels on λ- (and $\hat{\lambda}$-) abstractions.

References

[1] Iliano Cervesato and Frank Pfenning. A linear logical framework. In *Eleventh IEEE Symposium on Logic in Computer Science*, pages 264–275, July 1996.

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

[3] Karl Crary and Joseph C. Vanderwaart. An expressive, scalable type theory for certified code. Technical Report CMU-CS-01-113, Carnegie Mellon University, May 2001.

[4] Amy Felty. Encoding dependent types in intuitionistic logic. In Gérard Huet and Gordon D. Plotkin, editors, *Logical Frameworks*, pages 214–251. Cambridge University Press, 1991.

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

[6] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. Technical Report CMU-CS-99-159, Carnegie Mellon University, September 1999.

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

[8] Frank Pfenning. Personal communication.

[9] Jeff Polakow. *Ordered Linear Logic and Applications*. PhD thesis, Carnegie Mellon University, August 2001. Available as Technical Report CMU-CS-01-152.