Equality Between Functionals in the Presence of Coproducts

Daniel J. Dougherty   Ramesh Subrahmanyam*

Department of Mathematics
Wesleyan University
Middletown, CT 06459 USA

Abstract

We consider the lambda-calculus obtained from the simply-typed calculus by adding products, coproducts, and a terminal type. We prove the following theorem: The equations provable in this calculus are precisely those true in any set-theoretic model with an infinite base type.

1 Introduction

The model theory of the simply-typed lambda calculus, $\lambda^{\rightarrow}$, has been well developed in the last two decades. For the most part, techniques and results generalize readily to the calculus when product types are added. Indeed, a categorical treatment goes more smoothly in the presence of products. But very little is known about the model theory of the simply-typed lambda calculus with coproducts for two chief reasons. First, techniques in the model theory of $\lambda^{\rightarrow}$ often rely heavily on the strong syntactic properties of the calculus; many of these properties fail in the presence of coproducts. Second, the natural generalizations of several key theorems in the model theory of $\lambda^{\rightarrow}$ fail in the setting with coproducts (see Section 3). We conclude that new techniques must be developed to study the model theory of the lambda calculus with coproducts; this paper makes a start.

There is a natural candidate for an axiomatization of this calculus, obtained from $\lambda^{\rightarrow}$ by adding type-constructors for binary products and sums and a unit type, and term-constants suggested by the (equational) axiomatization of the theory of bi-cartesian categories. The presence of an initial type leads to severe difficulties in the syntactic analysis, and we will in this paper omit a treatment of the system with an initial type. We denote the resulting theory by ABC (for "almost bi-cartesian closed"); the defining equations are given in Table 2.

The structure we are primarily interested in is the set-theoretic type hierarchy Set, the family $\{Set^{\tau} \mid \tau \in Types\}$ obtained by taking Set^{ι} to be any infinite set and interpreting the type-constructors $\rightarrow$, $\times$, $+$, and $\mathbf{1}$ respectively as full function-space, cartesian product, disjoint sum and a singleton set. Of course this is a model for the theory; it may be viewed as the "intended model" when the lambda calculus is considered as a theory of functions.

*Supported by the National Science Foundation through Grant CCR-9406202.

Our main result is a strong completeness theorem, to the effect that that an equation is true in the set-theoretic model *Set* if and only if it is provable in the equational theory presented by the axioms ABC described above.

This generalizes the corresponding result for the $\lambda^{\rightarrow}$-calculus, obtained by Harvey Friedman in the seminal [Fri75]: it is proved there that equality between simply-typed lambda terms in the full function-type hierarchy over an infinite set is completely axiomatized by β and η.

There does not appear to be an equational presentation of the theory which supports a confluent rewrite system (cf Section 3) but the main technical result arising out of our proof analysis is the demonstration of a certain *consequence* (Theorem 7.13) of confluence in better-behaved calculi; this lemma leads to a method for encoding the negation of equations in the calculus (Proposition 8.1) which is the key to the completeness proof.

An important aspect of our approach is the use of a proof system consisting of the usual rules of equational reasoning extended by an explicit rule for reasoning by cases. This extension is shown to be sound in Section 5. The presence of such a rule means that reasoning with non-logical axioms is an essential part of our approach, and techniques from the theory of (first-order) term rewriting play an important role in the development. A central role is played by the method of rewriting modulo an equivalence relation [Hue80]; more about this in Section 7. Modifications to the standard theory are required in the presence of abstraction and β-conversion.

Related work The syntactic properties of the λ-calculus with coproduct (and/or weak coproduct) types have received a lot of attention recently. The systems typically studied have been variants of the equational theory of bi-cartesian-closed categories (see [LS86] for the relationship between equationally defined λ-calculi and cartesian-closed categories.). Both [DK93] and [Dou93] show that a theory axiomatizing *weak* coproducts and primitive recursive functionals at higher types is strongly normalizing and confluent; the latter paper additionally shows strong normalization for a theory with true coproducts. Okada and Scott [OS91] have presented a similar result for bi-cccs with a weak natural numbers object. We should also mention the work of Čubrić, [Cub92] who adapted Friedman's completeness theorem to show that there is a faithful ccc-functor from any free cartesian-closed category into the category of Sets.

Organization The paper is organized as follows. The syntax of our language is described in Section 2. Section 3 gives empirical evidence for the need for new techniques for addressing the problem at hand. Section 4 outlines the proof of main theorem, by way of motivating some of the technical notions to be developed in the body of the paper. Section 5 shows that the system for reasoning by cases is equivalent to the ABC theory. Sections 6 and 7 are devoted to an analysis of derivations. Section 8 gives the proof of the completeness theorem. The proofs of some technical results in Sections 5 and 6 are relegated to appendices.

2 Syntax

We require a somewhat delicate syntactic apparatus. We work with closed terms and so will postulate constants of each type, but we also need, for purely technical reasons, to maintain specific relationships between certain constants at the meta-level. For example, if c is a constant of product type $\sigma_1 \times \sigma_2$, then we want to have constants c_1 and c_2 of types σ_1 and σ_2 naming $\mathsf{pr}_1 c$ and $\mathsf{pr}_2 c$, where the pr_i are the canonical projection functions. A similar (but subtler) situation arises with sum-types. Since of course the σ_i types above might themselves be products or sums, some machinery is needed to do the bookkeeping.

Types and Terms The *base types* are $\mathbf{1}$ and ι. The set of *types* is the closure of the set of base types under the constructions $(\sigma \times \tau)$, $(\sigma + \tau)$, and $(\sigma \to \tau)$.

For each type σ let $Vars_\sigma$ be an infinite set of variables; we assume that the $Vars_\sigma$ are pairwise disjoint, and set $Vars$ to be be $\bigcup\{\, Vars_\sigma \mid \sigma \text{ a type } \}$.

Similarly, let Σ be a set of constants with infinitely many constants of each type. The set of *raw extended constants* is a set of pairs (c, ρ) where $c \in \Sigma$, and ρ is a sequence of 1's and 2's. Instead of the more cumbersome pair notation, we write c_ρ. We associate types with some of the raw extended constants as follows. The type of $c_{\langle\rangle}$ is the type of c in Σ. The type of $c_{\rho i}$ is $\nu_1 \to \ldots \to \nu_k \to \tau_i$ if the type of c_ρ is $\nu_1 \to \ldots \to \nu_k \to \tau_1 + \tau_2$. Clearly, not all raw extended constants receive types; the set of extended constants with type σ is denoted $\Sigma^\star_\sigma$. Then the set of all typable extended constants is denoted $\Sigma^\star$. There is an obvious injection from Σ to $\Sigma^\star$, namely $c \mapsto c_{\langle\rangle}$; in the following we will identify $c \in \Sigma$ with $c_{\langle\rangle} \in \Sigma^\star$. Small letters at the beginning of the English alphabet will usually be used to represent constants. If $c \in \Sigma^\star$ then $c \equiv d_\rho$, for some $d \in \Sigma$ and sequence of 1s and 2s ρ; then c_i, for instance, will denote the constant $d_{\rho i}$.

In addition to these symbols, we also have several type-indexed families of interpreted constants, namely,

- $\langle \cdot, \cdot \rangle^{\sigma_1,\sigma_2} \colon \sigma_1 \to \sigma_2 \to \sigma_1 \times \sigma_2$,
- $\mathsf{pr}_1^{\sigma_1,\sigma_2} \colon \sigma_1 \times \sigma_2 \to \sigma_1$ and $\mathsf{pr}_2^{\sigma_1,\sigma_2} \colon \sigma_1 \times \sigma_2 \to \sigma_2$,
- $\mathsf{in}_1 \sigma_1, \sigma_2 \colon \sigma_1 \to (\sigma_1 + \sigma_2)$ and $\mathsf{in}_2 \sigma_1, \sigma_2 \colon \sigma_2 \to (\sigma_1 + \sigma_2)$,
- $\mathsf{case}^{\sigma_1,\sigma_2,\tau} \colon (\sigma_1 + \sigma_2) \to (\sigma_1 \to \tau) \to (\sigma_2 \to \tau) \to \tau$, and
- $* : \mathbf{1}$.

In the rest of this paper, unless otherwise restricted, a constant will mean an element of the set $\Sigma^\star \cup \{\mathsf{pr}_i, \mathsf{in}_i, \mathsf{case}, *, \langle ., . \rangle\}$. We also have an infinite set of variables disjoint from the set of constants.

The raw terms of our calculus will consist of constants, variables, applications $(M\ N)$, and lambda abstractions $\lambda x^\tau.M$, where M and N are raw terms. Closed terms will refer to terms with no occurrences of free variables; note that such terms may contain symbols from $\Sigma^\star$. In the concrete syntax, parentheses will suppressed whenever possible (under the usual conventions that the function-space constructor associates right and term application associates left) and terms will be considered identical if they differ only by renaming of bound variables, and type information will be omitted if it can be easily inferred. Often type superscripts for λ-bound variables as well as the interpreted constants above will be omitted when they can be inferred from the context. The application $\langle .,. \rangle\ M\ N$ will be abbreviated $\langle M, N\rangle$. We use $M \equiv N$ to indicate that M and N are syntactically identical.

Terms containing no constants from Σ or $\Sigma^\star$ will be called *pure* terms.

The constants pr_i and case have a different character from the others in that they trigger reductions, as described below in Definition 6.2. Accordingly we refer to the constants among the in_i, $\langle\cdot,\cdot\rangle$, $*$, and the sets Σ and $\Sigma^\star$ as *passive constants*.

Certain raw terms are well-typed according to the usual typing rules for simply-typed lambda calculus; in the sequel, unless explicitly specified, a term will always mean a typable term. [See Table 1.]

It will often be convenient to use a "vector notation" for terms, in the following sense: if F is a term of type $\sigma_1 \to \cdots \to \sigma_n \to \tau$ then $F\vec{A}$ denotes the term of type τ obtained by applying terms $A_1, \ldots, A_n$ of appropriate type.

The substitution of term A for variable x in B is denoted $B[A/x]$.

Definition 2.1 A closed term is an *introduction*-term if it is of one of the forms:

$$\mathsf{in}_i(M), \qquad \langle M, N\rangle, \qquad *, \qquad \lambda x.N.$$

A term is *resolved* if each of its closed subterms is an introduction term.

A closed term of sum or product type is *rigid* if it is of the form $hA_1 \cdots A_n$ with $h \in \Sigma^\star$ and each A_i is resolved.

An *equation* is a pair of terms of the same type.

The theory ABC is the equational theory presented by the axioms of Table 2.

A set Γ of sentences is *consistent* if Γ does not entail, under ABC, the equation $x = y$.

The ABC equations for the sum ensure that the meaning of a term G of type $(\sigma_1 + \sigma_2) \to \tau$ is determined by its action on terms from σ_1 and from σ_2. This is the sense in which the sum type is "categorical"; a similar remark applies for product type.

3 Coproducts considered difficult

Here we give some indications of the ways that the calculus with coproducts differs from the simpler arrow- and product-type systems.

$$\frac{x^\sigma \in \mathit{Vars}_\sigma}{x{:}\,\sigma} \qquad \frac{c \in \Sigma^*_\sigma}{c{:}\,\sigma}$$

$$\frac{M{:}\,\tau}{\lambda x^\sigma.M{:}\,\sigma\to\tau} \qquad \frac{M{:}\,\sigma\to\tau \quad N{:}\,\sigma}{(MN){:}\,\tau}$$

Table 1: Terms

(β)	$(\lambda x.B)A$	$=$	$B[A/x]$
$(\times)$	$\mathsf{pr}_i\langle A_1, A_2\rangle$	$=$	$A_i, \quad i \in \{1,2\}$
(case)	$(\mathsf{case}\ \mathsf{in}_i A\ F_1\ F_2)$	$=$	$F_i A, \quad i \in \{1,2\}$
(η)	F	$=$	$\lambda x.Fx \qquad x$ not free in F
$(+!)$	h	$=$	$\lambda x.(\mathsf{case}\ x\ \lambda y.h(\mathsf{in}_1(y))\ \lambda z.h(\mathsf{in}_2(z)))$
$(\mathbf{1}!)$	U	$=$	$*$
$(\times!)$	P	$=$	$\langle \mathsf{pr}_1 P, \mathsf{pr}_2 P\rangle$

Table 2: The equational theory ABC

Strong extensionality

The set-theoretic model for function-types over an infinite base set satisfies the following *strong extensionality property*: If $\{f_1, \ldots, f_n\}$ is any finite set of λ-definable functions of the same type then there is a single element a which distinguishes them, that is, the elements $f_1(a), \ldots, f_n(a)$ are distinct. This may be seen by examining Friedman's proof of his completeness theorem.

This strong extensionality property fails in *Set* (as a model for the calculus with coproducts). Define:

$$G = \lambda xyz.(\mathsf{case}\ x\ \lambda u.y\ \lambda u.z)$$
$$F_1 = \lambda xyz.y,$$
$$F_2 = \lambda xyz.z$$

Then G differs from each of F_1 and F_2 in *Set* but no single input vector distinguishes all three of them. ///

To see the technical significance of this observation note that any open term model for a set of equations will be strongly extensional: if $\{F_1, \ldots, F_n\}$ is any finite set of terms which are pairwise not equal in the model, then any equivalence class containing a variable serves as a uniform witness to their inequality. But the reader familiar with logical relations will see that if Friedman's proof technique were transferable directly to a λ-calculus with coproducts, strong extensionality for *Set* would follow.

Confluence and decidability

For the λ-calculus with function and product types, reduction is confluent and strongly normalizing. Decidability follows.

But all of the usual axiomatizations of coproducts fail to be confluent. (Weak coproduct calculi, which omit the (+!)-rule of ABC tend to be confluent, as in [DK93] and [Dou93].) Strong normalization holds, but a confluent rewrite system which is uniform in the sense of being closed under substitution seems elusive. Indeed, we conjecture that no such system exists. A partial result along these lines is shown in [Dou93]: there cannot exist a *left-linear* rewrite system complete for an equational theory of coproducts with even modest expressive power (a system is left-linear if there are no repeated variables on the left-hand sides of rules.)

This suggests that showing decidability for coproduct theories, such as ABC and the theory of bi-ccc's, is difficult.

Statman's 1-section theorem

Let $\mathcal{C}$ be a class of models of the simply-typed lambda calculus $\lambda^{\rightarrow}$. Statman [Sta82] has shown that the equations valid in this class are completely axiomatized by β and η iff the free algebra of binary trees can be fully and faithfully embedded in some countable direct product of models in $\mathcal{C}$. This is known as the 1-section theorem.

Equivalently: consider the signature Σ with a constant c of base type and a constant p of type $\iota \rightarrow \iota \rightarrow \iota$; closed base-type terms over this signature can be naturally identified with binary trees. The 1-section theorem states that the equations true in $\mathcal{C}$ are axiomatized by β and η if for each natural number d, there is a model $\mathcal{A} \in \mathcal{C}$ and an expansion of $\mathcal{A}$ to the language Σ such that all closed base-type terms whose depth (as trees) is bounded by d have distinct interpretations.

A corollary of the theorem is the finite model theorem: if an equation is true in all finite models then it is true in all models.

The proof of the 1-section theorem relies on the following technical lemma: If two pure $\lambda^{\rightarrow}$-terms S and T are not equal under $\beta\eta$ then there is a context $C[\,]$ of base type containing only *first-order* constants such that $C[S]$ and $C[T]$ are not equal under β and η. Indeed, it suffices to have a single constant of base type and a single constant of type $\iota \rightarrow \iota \rightarrow \iota$.

As the following lemma shows, for the theory of *Set* the lemma fails, even if we relax "first-order" to read "hereditary Harrop." Harrop types are types of the form $\tau_1 \rightarrow \tau_2 \ldots \rightarrow \iota$. These types are those corresponding to Harrop formulas of propositional logic under the well-known propositions-as-types analogy. Hereditary Harrop types are those that have no occurrence of $+$.

Lemma 3.1 *There are terms M and N such that $Set \not\models M = N$ but for any context $C[]$ of base type whose free constants are of hereditary Harrop type $Set \models C[M] = C[N]$.*

Proof. Let *test* be the term

$$\lambda xyuv.(\mathsf{case}\ x\ (\ \mathsf{case}\ y\ u\ v)(\mathsf{case}\ y\ v\ u)),$$

of type $(\iota + \iota) \to (\iota + \iota) \to \iota \to \iota \to \iota$. This denotes the four-argument function which returns the third argument if either its first two arguments are both injected from the left or both injected from the right; otherwise it returns the fourth argument. Consider the terms $M \equiv \lambda fxyuv.test\ (f\ x)\ (f\ y)\ u\ v$ and $N \equiv \lambda fxyuv.u$, of type $\tau \equiv (\iota \to (\iota + \iota)) \to \iota \to \iota \to \iota \to \iota \to \iota$. These two terms are not equal in *Set*.

The reduction relation obtained by orienting β and $(case)$ from left-to-right is confluent and strongly normalizing. We will show by induction on the size of contexts $C[]$, which are in normal form with respect to this reduction, that $Set \models C[M] = C[N]$.

Let $C[]$ be a normal-form context of base type with hole of type τ and in which every free constant is of hereditary Harrop type. Analysis of the structure of $C[]$ reveals that it must be of the form $([]\ F[]\ X[]\ Y[]\ U_1[]\ U_2[])$ for normal-form contexts $F[], X[], Y[], U_1[]$, and $U_2[]$. It is easy to see that normal forms of terms of type $\iota \to (\iota + \iota)$, all of whose free constants are of Harrop types, have the shape $\lambda x.\mathsf{in}_i A$ where $i \in \{1, 2\}$. Noting that $[]$ is of Harrop type, and applying this observation to $F[]$, we conclude that $F[] \equiv \lambda x.\mathsf{in}_i F'[]$, for some $i \in \{1, 2\}$ and for some context $F'[]$. Thus $Set \models C[M] = U_i[M]$ and $Set \models C[N] = U_i[N]$. $U_i[]$ is a smaller context and, so, using the induction hypothesis $Set \models U_i[M] = U_i[N]$ we get $Set \models C[M] = C[N]$. ///

The natural generalization of the 1-section theorem to the calculus with coproducts also fails. The failure of the 1-section theorem and the key technical lemma indicates that a new proof technique must be explored.

4 The disjunction property

A *typed applicative structure,* henceforth simply *structure,* is a family $\mathcal{A}$ of non-empty sets $\mathcal{A}^\tau$ indexed by the types, with binary operations playing the role of application-operators. A *model* is a combinatorially complete extensional applicative structure. See [Bar84] for a detailed treatment in the untyped case. Of course *Set* is a model in a natural way.

The usual term-model construction yields an abstract completeness theorem: Δ has a model iff Δ is consistent. Indeed, the set of terms $\mathcal{T}$ may be considered as a structure, with juxtaposition as application. If Δ is a set of closed equations then the relation $\Delta \vdash M = N$ induces a congruence relation on terms, and in the usual manner induces a quotient structure $\mathcal{T}_\Delta$. When the language has infinitely many constants of each type, then $\mathcal{T}_\Delta$ is a model satisfying all of the equations in Δ, the *term model* for Δ.

Friedman's proof of the completeness theorem for $\lambda^\to$ may be described as follows: Let $\mathcal{T}$ be the closed term model for the empty theory in a language with infinitely many constants of each type. If we define an arbitrary injective function from the base type of $\mathcal{T}$ to the base type of $Set^\to$ (recall that $Set^\to$ refers to the function hierarchy over an infinite base set) then this function lifts uniquely to a total injective *logical relation* [Mit90] from $\mathcal{T}$ to *Set*. Since logical relations are guaranteed to relate the meanings of terms, we conclude that equations true in $Set^{\rightarrow}$ are provable.

This argument does not generalize easily to the setting with coproducts; in this section we isolate the key obstacle. Here is a natural definition of logical relation in our setting:

Definition 4.1 Let $\mathcal{M}$ and $\mathcal{N}$ be models. A *logical relation* $\mathcal{R}$ from $\mathcal{M}$ to $\mathcal{N}$ is a family of relations $\mathcal{R}^\sigma \subseteq \mathcal{M}^\sigma \times \mathcal{N}^\sigma$ indexed by types, satisfying:

- $\mathcal{R}^{\sigma_1 \rightarrow \sigma_2}(m, n)$ iff for every a and b such that $\mathcal{R}^{\sigma_1}(a, b)$ we have $\mathcal{R}^{\sigma_2}(ma, nb)$;
- $\mathcal{R}^{\sigma_1 \times \sigma_2}(m, n)$ iff $\mathcal{R}^{\sigma_i}(\mathsf{pr}_i m, \mathsf{pr}_i n)$, for $i = 1, 2$;
- $\mathcal{R}^{\mathbf{1}}$ holds between the respective (single) elements at type $\mathbf{1}$;
- $\mathcal{R}^{\sigma_1 + \sigma_2}(m, n)$ iff there exists i such that $m = \mathsf{in}_i a$, $n = \mathsf{in}_i b$, and $\mathcal{R}^{\sigma_i}(a, b)$.

The "fundamental theorem of logical relations" is the following

Theorem 4.2 *If $\mathcal{R}$ is a logical relation between models $\mathcal{M}$ and $\mathcal{N}$, then for any pure closed lambda term T, $\mathcal{R}$ relates the meanings of T in the two models.*

Proof. The proof is an easy modification of the standard proof for the $\lambda^{\rightarrow}$ calculus (see, for example, [Mit90]) and is omitted here. ///

In attempting to prove the completeness theorem for the coproduct calculus, we are thus led to try to construct an appropriate logical relation from the term model (for the empty theory) to *Set*. But one gets stuck; we see that a special property of models is required to make the construction go through:

Definition 4.3 A model $\mathcal{A}$ has the *disjunction property* if for each type $\tau_1 + \tau_2$ and each element $a \in \mathcal{A}^{\tau_1 + \tau_2}$ there is an i and an a_i such that $a = \mathsf{in}_i a_i$ in $\mathcal{A}$.

We will refer to such an $\mathcal{A}$ as a *d.p. model.*

Proposition 4.4 *Suppose M and N are pure closed terms and that $Set \models M = N$. Then $M = N$ holds in all d.p. models with countable base types.*

Proof. Let $\mathcal{D}$ be a d.p. model with $\mathcal{D}^\iota$ countable. Let $\mathcal{R}^0$ be a (total) injective function from $\mathcal{D}^\iota$ to Set^ι; this uniquely extends to a logical relation. Argue simultaneously by induction on types that each $\mathcal{R}^\sigma$ has domain all of $\mathcal{D}^\sigma$ and is injective. Showing that it is injective at σ will use the fact that it is total at lower types, and proving that it is total at sum types will require the disjunction property.

The theorem follows from the fundamental theorem on logical relations. ///

So it suffices to establish a refinement of the standard abstract completeness theorem, as follows.

> If $\not\vdash M = N$ then there is a d.p. model $\mathcal{A}$ with $\mathcal{A}^\iota$ countable in which $M = N$ fails.

Now, it is not hard to see that when the set Σ of constants is empty, then every *closed* term of sum-type already reduces, under $(\beta), (\eta)$, and $(case)$, to a term of the form $\mathsf{in}_i X$. So it is tempting to work with the structure of *closed* terms modulo $(\beta), (\eta)$, and $(case)$-provability; but this is not extensional unless there are enough constants, and so is not, in general, a model.

Our first step is to build Σ so that it has infinitely many constants of each type. Now when constants at sum-types are added the disjunction property for closed terms is obviously lost: a constant c of type $\sigma_1 + \sigma_2$ is not provably equal to any term of the form $\mathsf{in}_i T$. We will build our d.p. model as a term model for a set of equations which collectively will entail that each closed term of sum-type is equal to a term which is resolved in the sense of Definition 2.1.

But then extensionality is problematic again: we must ensure that if the new equations arrange that two arrow-type terms agree on all arguments then they are declared equal in the theory. So we somehow need to add constants to resolve terms, while ensuring extensionality as we go. The construction will occupy the coming sections.

As a consequence of our construction we are led to work with what should be considered a syntactic version of Kripke-style logical relations ([Plo80, MM91]). The similarity with techniques from intuitionistic semantics is striking.

5 The Theories ABC and BCT

As discussed in the previous section, the axioms of ABC will not imply that all elements inhabiting a sum-type must in fact be of the form $\mathsf{in}_i x$. For example, the (open) term model for the axioms has variables of type $\sigma_1 + \sigma_2$. But, perhaps surprisingly, the equations above do support a principle of "reasoning by cases" by which we may infer an equation by proving it under the additional hypothesis that a given term of sum-type is of the form $\mathsf{in}_i x_i$, for $i = 1$ and 2.

The most straightforward formalization of this idea is a sequent calculus for deriving equations under hypotheses, with rules involving assumption-equations of the form $Q = \mathsf{in}_i c_i$ for closed terms Q and "fresh" constants c_i. It is convenient, though, to go a step further. The terms of sum-type which will not be provably equal to any $\mathsf{in}_i X$ will be, as it turns out, those of the form $f\vec{A}$ for some $f{:}\,\vec{\sigma} \to (\tau_1 + \tau_2)$. Now, when all types are inhabited there are definable functions $\mathsf{out}_i{:}\,(\tau_1 + \tau_2) \to \tau_i$:

$$\mathsf{out}_1 \equiv \lambda x^{\tau_1+\tau_2}.(\mathsf{case}\ x\ (\lambda y^{\tau_1}.y)\ T)$$

Axioms

$$
\begin{array}{llcl}
(\beta) & (\lambda x.B)A & = & B[A/x] \\
(case) & (\mathsf{case}\ \mathsf{in}_i A\ F_1\ F_2) & = & (F_i A) \\
(\eta) & F & = & (\lambda x.Fx), x \text{ not free in } F \\
(\times!) & (f\vec{A}) & = & \langle f_1\vec{A}, f_2\vec{A}\rangle, f \in \Sigma^\star
\end{array}
\qquad
\begin{array}{llcl}
(\pi) & \mathsf{pr}_i\langle M_1, M_2\rangle & = & M_i \\
\\
\\
(\mathbf{1}!) & M & = & *
\end{array}
$$

Rules

$$\frac{e \in \Delta \text{ or } e \text{ is an axiom}}{\Sigma_0; \Delta \rhd e} \qquad \frac{}{\Sigma_0; \Delta \rhd M = M}$$

$$\frac{\Sigma_0; \Delta \rhd M = N}{\Sigma_0; \Delta \rhd N = M} \qquad \frac{\Sigma_0; \Delta \rhd M = N \qquad \Sigma_0; \Delta \rhd N = P}{\Sigma_0; \Delta \rhd M = P}$$

$$\frac{\Sigma_0; \Delta \rhd F = F' \qquad \Sigma_0; \Delta \rhd A = A'}{\Sigma_0; \Delta \rhd FA = F'A'} \qquad (\xi)\ \frac{\Sigma_0 \cup \{c\}; \Delta \rhd U[c/x] = V[c/x]}{\Sigma_0; \Delta \rhd \lambda x.U = \lambda x.V}\ c \in \Sigma - \Sigma_0$$

$$(\text{ByCases})\ \frac{{\Sigma_0}^+; \Delta, h\vec{P} = \mathsf{in}_1(h_1\vec{P}) \rhd M = N \qquad {\Sigma_0}^+; \Delta, h\vec{P} = \mathsf{in}_2(h_2\vec{P}) \rhd M = N}{{\Sigma_0}^+; \Delta \rhd M = N}$$

In ByCases, ${\Sigma_0}^+$ is $\Sigma_0 \cup \{c \in \Sigma \mid c \text{ appears in } h\vec{P}\}$

Table 3: the sequent calculus BCT

$$\mathsf{out}_2 \equiv \lambda x^{\tau_1+\tau_2}.(\mathsf{case}\ x\ T\ (\lambda y^{\tau_2}.y))$$

where in each case T is some fixed term of the appropriate type. If an element of type $\sigma_1 + \sigma_2$ is of the form $\mathsf{in}_i X$ then out_i will extract the X. So in the above situation, out_i composed with f is a map from $\vec{\sigma}$ to τ_i, and so to postulate that $f\vec{A}$ is in the range of in_i is to say that it is precisely $\mathsf{in}_i(\mathsf{out}_i(f\vec{A}))$. The technical development below is smoothed by associating specific constants f_1 and f_2 to play the roles of the $\lambda\vec{x}.\mathsf{out}_i(f\ \vec{x})$. This is supported by the structure imposed on constants described in the paragraph beginning this section.

This should motivate the rules of the sequent calculus BCT, the "By Cases Theory" given in the next definition. In Table 3 we use the vector-notation described earlier: in rule $(\times!)$, the terms $\vec{A}$ are of types such that $(f\vec{A})$ has product type; and in rule ByCases, the terms $\vec{P}$ are of types such that $(h\vec{P})$ has sum type.

Definition 5.1 A *sequent* is a triple, $\Sigma_0; \Delta \rhd e$, where $\Sigma_0 \subseteq \Sigma$, Δ is a finite set of equations, and e is an equation, such that if c_ρ appears in Δ or in e then $c \in \Sigma_0$.

The calculus BCT is the system for deriving sequents which is described in Table 3. A side condition applying to every rule is that the premises and conclusions are sequents; specifically, in $\Sigma_0; \Delta \rhd e$, Σ_0 contains all the constants that appear in Δ and e.

We write $\Delta \vdash_{\mathsf{BCT}} M = N$ if there is a proof of the sequent $\Sigma_0; \Delta \rhd M = N$ for some Σ_0.

As suggested above, the intuition behind the ByCases rule as follows. Suppose we want to derive an equation $M = N$ under certain hypotheses, and suppose that $h\vec{P}$ is some term of sum-type. In BCT, we may conclude $M = N$ if we can derive it under the additional hypothesis that $h\vec{P}$ is an injection from the left, and also under the hypothesis that $h\vec{P}$ is an injection from the right.

The reader will observe that the standard surjective pairing axiom $(\times!)$ is postulated only for terms with an extended constant at the head. This convention smooths the rewriting-style technical development to come; it is a consequence of the main completeness theorem that this is no restriction on the proof theory.

In this section we show that BCT is equivalent to the theory ABC when the set of hypotheses is empty. This is shown as follows: we first define a variant of BCT called WBCT, the "weak" By Cases theory, and prove that WBCT proves all pure equations that BCT does. We next show that ABC is as strong as WBCT. It is clear that ABC is sound, as all of its equations are true in *Set*; since BCT will be shown by the completeness theorem to be complete for *Set*, we conclude that all three systems have the same proof strength with respect to equations between pure closed terms.

5.1 The theory WBCT

Definition 5.2 We now examine a slightly different proof system called WBCT. This is obtained by making the following changes to the language and rules of BCT. First $\Sigma^\star$ is dispensed with, and thus the only constants under consideration are Σ and the interpreted constants. Hypotheses are of the form $Q = \mathsf{in}_i(c)$, where Q is a term of sum type, and $c \in \Sigma$.

Secondly the ByCases rule of BCT is replaced by the following rule:

$$\frac{{\Sigma_0}^l; \Gamma, Q = \mathsf{in}_1(c) \triangleright M = N \qquad {\Sigma_0}^r; \Gamma, Q = \mathsf{in}_2(c') \triangleright M = N}{\Sigma_0; \Gamma \triangleright M = N}$$

where $c, c' \in \Sigma$ do not occur in the left and right premises, respectively, ${\Sigma_0}^l = \Sigma_0 \cup \{c\} \cup \{c \mid c \text{ appears in } Q\}$ and ${\Sigma_0}^r = \Sigma_0 \cup \{c'\} \cup \{c \mid c \text{ appears in } Q\}$

Finally, the axiom $(\times!)$ is replaced with the usual surjective pairing equation: $M = \langle \mathsf{pr}_1 M, \mathsf{pr}_2 M\rangle$.

In this subsection we show an interpretation of BCT in WBCT. Given a sequent to translate, choose $C \subseteq \Sigma$ such that the constants in C do not appear in the sequent, and let $d \in \Sigma$ be a fresh constant of type ι. Using d we may define a collection of terms T^σ for each type σ. The function $\mathcal{E}$ is defined on $\Sigma^\star$ as follows:

$$\mathcal{E}(f) = \; f \quad \text{if } f \in C \ ,$$

$$\mathcal{E}(f_{\rho i}) = \begin{cases} \lambda\vec{x}.(\mathsf{pr}_i(\mathcal{E}(f_\rho)\vec{x})) & f\colon \tau_1 \to \cdots \to (\nu_1 \times \nu_2) \\ \lambda\vec{x}.(\mathsf{out}_i(\mathcal{E}(f_\rho)\vec{x})) & f\colon \tau_1 \to \cdots \to (\nu_1 + \nu_2) \end{cases} .$$

The function can be extended to terms by treating it as a substitution. Thus $\mathcal{E}(M)$ is obtained by replacing, for each $f_\rho \in \Sigma^\star$, every instance of f_ρ by $\mathcal{E}(f_\rho)$.

For set of hypotheses Δ, define

$$\mathcal{E}(\Delta) = \{\mathcal{E}(f_\rho \vec{M}) = \mathsf{in}_i(c_{(f_\rho \vec{M})}) \mid f_\rho \vec{M} = \mathsf{in}_i(f_{\rho i} \vec{M}) \in \Delta\}.$$

$\mathcal{E}$ may now be extended to sequents; $\mathcal{E}(\Sigma_0; \Delta \triangleright M = N)$ is ${\Sigma_0}^+; \mathcal{E}(\Delta) \triangleright \mathcal{E}(M) = \mathcal{E}(N)$, where ${\Sigma_0}^+$ is Σ expanded to include all the constants that appear in $\mathcal{E}(\Delta)$.

Theorem 5.3 *If $\Sigma_0; \Delta \triangleright M = N$ is provable in* BCT *then its translation $\mathcal{E}(\Sigma_0; \Delta \triangleright M = N)$ is provable in* WBCT.

Proof. The proof proceeds by induction on the structure of derivations, and is routine. Two representative cases are described.

- The last step in the deduction is an instance of the ξ-rule.

  $$\frac{\Sigma_0 \cup \{d\}; \Delta \triangleright M[d/x] = N[d/x]}{\Sigma_0; \Delta \triangleright \lambda x.M = \lambda x.N} \quad d \in \Sigma - \Sigma_0$$

  By induction hypothesis, ${\Sigma_0}^+ \cup \{d\}; \mathcal{E}(\Delta) \triangleright \mathcal{E}(M[d/x]) = \mathcal{E}(N[d/x])$ is derivable in WBCT. Since $d \in \Sigma$ but $d \notin \Sigma^\star$, $\mathcal{E}(M[d/x]) \equiv \mathcal{E}(M)[d/x]$. By a use of the ξ-rule in WBCT, conclude $\mathcal{E}(\Sigma_0; \Delta \triangleright \lambda x.M = \lambda x.N)$.

- $\Sigma_0; \Delta \triangleright M = N$ since $M = N \in \Delta$. Suppose $M = f_\rho \vec{M}$ and $N = \mathsf{in}_i(f_{\rho i} \vec{M})$. Then

  $$\begin{aligned}
  \mathcal{E}(N) &\equiv \mathsf{in}_i(\mathcal{E}(f_{\rho i} \mathcal{E}(\vec{M}))) \\
  &= \mathsf{in}_i(\mathsf{out}_i(\mathcal{E}(f_\rho)\mathcal{E}(\vec{M}))) \\
  &= \mathsf{in}_i(\mathsf{out}_i(\mathsf{in}_i(c_{f_{\rho i}\mathcal{E}(\vec{M})}))) \\
  &= (\mathsf{in}_i(c_{f_{\rho i}\mathcal{E}(\vec{M})})) \\
  &= \mathcal{E}(M).
  \end{aligned}$$

  Note the two uses of the axiom $\mathcal{E}(f_\rho \vec{M}) = \mathsf{in}_i(c_{(f_\rho \vec{M})}) \in \mathcal{E}(\Delta)$ in the above proof.

///

5.2 WBCT and ABC

In this subsection we give a translation of sequents from WBCT to closed constant-free equations over the language defined so far. In particular, when M and N are pure the translation will map the judgment $\emptyset; \emptyset \triangleright M = N$, to the eqaution $M = N$. We will show that the translation maps derivable sequents in WBCT to equations provable in the theory ABC.

Say that a list Δ of hypotheses is *proper* if each hypothesis is of the form $t_1 = \mathsf{in}_{\delta_1} a_1, \ldots$, is closed (perhaps containing constants) and finally, for every j, a_j does not appear in any of the first $j-1$ equations. In a proof of an equation in the ByCases theory, if the hypotheses are maintained as lists and the ByCases rule is written as follows

$$\text{(ByCases)} \quad \frac{{\Sigma_0}^l; \Delta, Q = \mathsf{in}_1(a_1) \ \triangleright\ M = N \qquad {\Sigma_0}^r; \Delta, Q = \mathsf{in}_2(a_2) \ \triangleright\ M = N}{\Sigma_0; \Delta \ \triangleright\ M = N}$$

then it is easy to see that all hypotheses-lists that appear in a proof of an equation starting with empty hypotheses are proper.

With every proper hypotheses-list Δ and sequence Υ of λ-terms of the same length we can associate a context $\widehat{\Delta}_\Upsilon[]$ defined as follows:

$$\widehat{\Delta}_\Upsilon[] \quad = \quad \begin{cases} [] & \textit{if } \Delta = [] \\ \mathsf{case}\ t_1\ \lambda a_1.(\widehat{\Delta^-}_{\Upsilon^-}[])\ \Upsilon(0) & \text{where } \Delta = (t_1 = \mathsf{in}_1(a_1)) \cdot \Delta^-,\ \Upsilon = \Upsilon(0) \cdot \Upsilon^- \\ \mathsf{case}\ t_1\ \Upsilon(0)\ \lambda b_1.(\widehat{\Delta^-}_{\Upsilon^-}[]) & \text{where } \Delta = (t_1 = \mathsf{in}_2(b_1)) \cdot \Delta^-,\ \Upsilon = \Upsilon(0) \cdot \Upsilon^- \end{cases}$$

The terms above of course will not be well-typed for arbitrary Υ. However, in the lemmas and proofs below, we will assume that the Υ is always chosen so that the terms mentioned in the context are type correct. So, when a proposition asserts, *for all* Υ ... it should be understood that the quantification is always over those Υ for which the terms involved type check.

The following facts are proved in Appendix A.

> **Lemma** A.2. If for every Υ, $\widehat{\Delta}_\Upsilon[M] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[N]$ and for every Υ, $\widehat{\Delta}_\Upsilon[P] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[Q]$ then for every Υ $\widehat{\Delta}_\Upsilon[M\ P] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[N\ Q]$.
>
> **Lemma** A.4. If for every Υ, $\widehat{\Delta}_\Upsilon[M] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[N]$ then for every Υ, $\widehat{\Delta}_\Upsilon[\lambda x.M^x_c] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[N^x_c]$, provided $c, x \notin \Delta$ and x is not free in M and N.
>
> **Lemma** A.5. Let $\Delta^1 = \Delta.(t = \mathsf{in}_1(a))$ and $\Delta^2 = \Delta.(t = \mathsf{in}_2(b))$. If for every Υ_1, $\widehat{\Delta^1}_{\Upsilon_1}[M] =_{\mathsf{ABC}} \widehat{\Delta^1}_{\Upsilon_1}[N]$ and for every Υ_2, $\widehat{\Delta^2}_{\Upsilon_2}[M] =_{\mathsf{ABC}} \widehat{\Delta^2}_{\Upsilon_2}[N]$, and $a, b \notin M, N$ then for every Υ, $\widehat{\Delta}_\Upsilon[M] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[N]$.
>
> **Lemma** A.6. Let $\Delta = t_1 = \mathsf{in}_{\delta_1}(a_1), \ldots, t_j = \mathsf{in}_{\delta_j}(a_j), \ldots, t_n = \mathsf{in}_{\delta_n}(a_n)$. Then, for any Υ and any context $C[]$,
>
> $$\widehat{\Delta}_\Upsilon[C[t_j]] =_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[C[\mathsf{in}_{\delta_j}(a_j)]].$$

Theorem 5.4 $\Delta \vdash_{\mathsf{BCT}} M = N$ *iff* $\vdash_{\mathsf{ABC}} \widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[N]$.

Proof. Let $\Delta \vdash M = N$ in the reasoning-by-cases system. We will induct on the height of the proof tree. There are three cases corresponding to height $= 1$.

- $M \equiv N$. In this case $\widehat{\Delta}_\Upsilon[M] \equiv \widehat{\Delta}_\Upsilon[N]$.
- $M \equiv t_j$, $N \equiv \mathsf{in}_{\delta_j}(a_j)$, and $t_j = \mathsf{in}_{\delta_j}(a_j)$ is in Δ. With $C[] = []$ apply Lemma A.6.
- $M = N$ is an axiom in WBCT and hence in ABC.

When $h > 1$, the last rule in the proof could be any of the following.

- Symmetry. By induction hypothesis $\widehat{\Delta}_\Upsilon[N] = \widehat{\Delta}_\Upsilon[M]$ is provable. Therefore by symmetry $\widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[N]$ is provable.
- Transitivity. By IH $\widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[P]$ and $\widehat{\Delta}_\Upsilon[P] = \widehat{\Delta}_\Upsilon[N]$ are provable. Use transitivity.
- Congruence with respect to Applications. Let $M \equiv (P\ Q)$ and $N \equiv (R\ S)$, and $\Delta \vdash P = R$ and $\Delta \vdash Q = S$. By IH, $\widehat{\Delta}_\Upsilon[P] = \widehat{\Delta}_\Upsilon[R]$ and $\widehat{\Delta}_\Upsilon[Q] = \widehat{\Delta}_\Upsilon[S]$, for every Υ.Now use Lemma A.2.
- ξ-rule. Say $M \equiv \lambda x.U$, $N \equiv \lambda x.V$, and $\Delta \vdash U^c_x = V^c_x$, where c is fresh with respect to Δ, M and N. By IH, $\widehat{\Delta}_\Upsilon[U^c_x] = \widehat{\Delta}_\Upsilon[V^c_x]$ is provable, for every Υ. Invoke Lemma A.4.
- ByCases rule. By IH, $\Delta^1 \equiv \Delta, t = \mathsf{in}_1(a) \vdash M = N$ and $\Delta^2 \equiv \Delta, t = \mathsf{in}_2(b) \vdash M = N$. By IH, for every Υ, $\widehat{\Delta^1}_\Upsilon[M] = \widehat{\Delta^1}_\Upsilon[N]$ and $\widehat{\Delta^2}_\Upsilon[M] = \widehat{\Delta^2}_\Upsilon[N]$ are provable in the empty theory. Apply Lemma A.5. ///

6 Valuations and reductions

Definition 6.1 A *resolution equation* is an equation of the form

$$hA_1 \cdots A_n = \mathsf{in}_i(h_i A_1 \cdots A_n)$$

with a rigid left-hand side.

A *valuation* is a set of resolution equations in which no two equations have the same left-hand-sides.

Definition 6.2 1. *Weak reduction*, denoted $\xrightarrow{w}$, is the reduction relation obtained by orienting the ABC equations $(\beta), (\eta), (\times), (\mathbf{1}!)$, and $(case)$ from from left to right, subject to the following provisions:

- In (η)-reduction, the redex F is not already of the form $\lambda x.B$, and furthermore does not occur in a context (FA);
- In $(\mathbf{1}!)$-reduction, the redex is not the term $*$.

2. Let Γ be a valuation. Γ-*reduction*, denoted $\xrightarrow{\Gamma}$, is the reduction obtained by orienting the following equations from left to right:

$$fA_1 \cdots A_n = \langle f_1 A_1 \cdots A_n, f_2 A_1 \cdots A_n \rangle$$

whenever $fA_1 \cdots A_n$ has product type, and $f \in \Sigma^\star$, and

$$fA_1 \cdots A_n = \mathsf{in}_i(f_i A_1 \cdots A_n), \quad i \in \{1,2\},$$

when this equation is in Γ. Here of course $fA_1 \cdots A_n$ has sum type, and $f \in \Sigma^\star$.

Let $\xrightarrow{\Gamma w}$ denote $(\xrightarrow{\Gamma} \cup \xrightarrow{w})$.

Before proving the key properties of Γ-reduction it is convenient to define the following notion of reduction; it will be useful below in analyzing syntactic behavior of terms. A perusal of the rules of the BCT calculus with an eye toward backward-proof search also should motivate these rules.

We use explcit contexts to define this relation since, in contrast to traditional reduction-relations, the rules may *not* be applied in arbitrary contexts.

Definition 6.3 Let $\Longrightarrow$ be the reduction relation obtained by adding the following rules to those for weak reduction.

- $C[f\vec{A}] \Longrightarrow C[\mathsf{in}_i(f_i\vec{A})]$, where $f \in \Sigma^\star$, $i \in \{1,2\}$, $C[]$ is a context, and $f\vec{A}$ is a closed term of sum type.
- $C[f\vec{A}] \Longrightarrow C[\langle f_1\vec{A}, f_2\vec{A}\rangle]$, where $f \in \Sigma^\star$, $i \in \{1,2\}$, $C[]$ is a context, and $f\vec{A}$ is a closed term of product type.
- *Add arguments.* A term $\lambda x.M$ of arrow-type may be replaced by $M[x := d]$ where $d \in \Sigma$ is of the appropriate type.
- *Selection.* A term $hA_1 \cdots A_n$, with h any constant excepting pr_1 and pr_2, may be replaced by one of the A_i.

Observe that the latter two rules for $\Longrightarrow$-reduction must be applied at the top-level only, since they change the types of terms.

Theorem 6.4 *The reduction* $\Longrightarrow$ *is strongly normalizing.*

Proof. See Appendix B. ///

Theorem 6.5 *For any valuation* Γ*, the reduction* $\xrightarrow{\Gamma w}$ *is strongly normalizing and confluent.*

Proof. Strong normalization for $\xrightarrow{\Gamma w}$ follows from the fact that $\xrightarrow{\Gamma w}$ is contained in $\Longrightarrow$. Confluence then follows from local confluence, which is easily verified. ///

So it makes sense to refer to the $\xrightarrow{\Gamma w}$-normal form of a term M, which we will denote M_Γ. The notion of valuations turns out to be a key concept enjoying a number of important properties that cannot be expected from arbitrary sets of hypothesis. It will become clear, through Theorems 7.13 and 8.3, that valuations are "tame" hypotheses.

Say that a valuation Γ is *full* if for every closed sum-type term Q, Q_Γ is resolved. If Γ is a valuation we will say that Γ^* is a "full extension" of Γ if Γ^* is a valuation, $\Gamma \subseteq \Gamma^*$, and Γ^* is full.

We must note that provability in BCT, say from hypotheses Γ, is not captured by Γw-convertibility because of the presence of the ByCases rule. The next step is to ameliorate this situation.

Definition 6.6 Let Γ be a finite valuation. A *bar* for Γ is a set $\mathcal{B}$ of finite valuations that are supersets of Γ such that for each full valuation $\Gamma^* \supseteq \Gamma$ there is a $\Gamma' \in \mathcal{B}$ with $\Gamma' \subseteq \Gamma^*$.

The binary relation $\preceq$ on bars for a fixed valuation is defined as follows: $\mathcal{B}_1 \preceq \mathcal{B}_2$ if and only if $\forall \Gamma \in \mathcal{B}_2, \exists \Gamma' \in \mathcal{B}_1 . \Gamma' \subseteq \Gamma$.

Observe that a bar, just as any set of sentences, may or may not be consistent: we will typically need to take some care that bars we construct are consistent.

Lemma 6.7

- *The relation $\preceq$ is a preorder with finite meets and joins. If $\mathcal{B}_1$ and $\mathcal{B}_2$ are bars for a valuation Γ, then their meet $\mathcal{B}_1 \sqcap \mathcal{B}_2$ is $\mathcal{B}_1 \cup \mathcal{B}_2$, and their join $\mathcal{B}_1 \sqcup \mathcal{B}_2$ is $\{\Gamma_1 \cup \Gamma_2 \mid \Gamma_1 \in \mathcal{B}_1, \Gamma_2 \in \mathcal{B}_2,$ and there exists a full valuation $\bar{\Gamma}$, with $\Gamma_1 \cup \Gamma_2 \subseteq \bar{\Gamma}\}$.*
- *If $\mathcal{B}_1$ and $\mathcal{B}_2$ are bars for a valuation Γ, resolving terms M_1 and M_2 respectively then $\mathcal{B}_1 \sqcup \mathcal{B}_2$ is a bar for Γ resolving both M_1 and M_2. Furthermore, if $\Gamma' = \Gamma_1 \cup \Gamma_2$ is a consistent element of $\mathcal{B}_1 \sqcup \mathcal{B}_2$, then $(M_i)_{\Gamma'} \equiv (M_i)_{\Gamma_i}$*

Proof. The first fact follows easily from the definition of $\preceq$. To see the second fact, first note that if Γ resolves a term so does any superset of Γ, and if the two valuations are each consistent then they reduce the term to the same normal form; the rest follows by recalling that any element in $\mathcal{B}_1 \sqcup \mathcal{B}_2$ is greater than some element in $\mathcal{B}_1$, and this element is given to resolve M and N. ///

7 Analyzing Provability

This section contains the main technical development supporting the proof of the completeness theorem. In case the reader would like to skip the details on a first reading, we summarize: The discussion through Definition 7.1 is essential; Notation 7.5 is used frequently; and Theorem 7.11 is the major result concerning the analysis of derivations themselves. Theorem 7.13 is the key result needed for the completeness proof.

7.1 A normal form for derivations

Perhaps the first thing that occurs to someone considering a system with a rule like our ByCases is to analyze terms by expanding all the cases at the start: that is, test an equation by considering all the sum-type subterms appearing (say in the w-normal form of the terms) and building a tree of possibilities for the assumptions that these subterms denote elements injected from the left or the right, respectively. This amounts to pushing occurrences of the ByCases rule down to the bottom of a proof-tree. Indeed, the completeness of such a strategy is easy to verify with the following exception: Instances of ByCases cannot, in general, be permuted below the ξ-rule.

So without loss of generality we may see derivations as being stratified into *layers* separated by ξ-instances, with each layer having all instances of ByCases at the bottom (i.e., at the beginning of a backwards proof-search), and so lending itself to a rewriting-style analysis. Of course the set of hypotheses defining the rewrite system is different in each layer.

It is convenient to distinguish derivations according to the number of "levels" occurring, that is, the depth of nesting of (ξ)-inferences in a derivation. So write

$$\Gamma \vdash_n M = N$$

if there is a derivation of the sequent $\Sigma_0; \Gamma_0 \triangleright M = N$ with no more than n uses of (ξ) for some $\Gamma_0 \subseteq \Gamma$ and $\Sigma_0 \subseteq \Sigma$.

The next relation is, intuitively, that which holds between terms at the boundary between the layers.

Definition 7.1 When Γ is a valuation we define the family of relations $M \sim_n^\Gamma N$ for $n \in \mathbf{N}$ inductively as follows.

$$M \sim_n^\Gamma N, \text{ if } \Gamma \text{ is inconsistent}$$

$$M \sim_n^\Gamma M$$

$$\frac{F \sim_n^\Gamma F' \qquad A \sim_n^\Gamma A'}{FA \sim_n^\Gamma F'A'}$$

$$\frac{\Gamma_0 \vdash_{n-1} U[c/x] = V[c/x]}{\lambda x.U \sim_n^\Gamma \lambda x.V} \quad , \quad \text{if } \Gamma_0 \subseteq \Gamma \text{ and } c \text{ does not occur in } \Gamma_0$$

Lemma 7.2

1. *if* $M \sim_n^\Gamma N$ *and* $\Gamma \subseteq \Gamma'$ *then* $M \sim_n^{\Gamma'} N$.
2. *if* $M \sim_n^\Gamma N$ *and* Γ *is consistent then* $\Gamma \vdash_{n-1} M = N$

3. *if* $M \sim_n^\Gamma N$ *then for any* P *and variable* x*,* $M[P/x] \sim_n^\Gamma N[P/x]$*.*

4. *Assume* Γ *is consistent.*

   - *If* $FA \sim_n^\Gamma N$ *holds, then* N *must be of the form* $F'A'$*, with* $F \sim_n^\Gamma F'$ *and* $A \sim_n^\Gamma A'$*.*
   - *If* $\lambda x.B \sim_n^\Gamma N$ *holds, then* N *must be of the form* $\lambda x.B'$*, with* $\Gamma \vdash_{n-1} B[d/x] = B'[d/x]$ *for all constants* d*.*

Proof. These are all easy inductions over the definition of the $\sim$-relation. ///

Lemma 7.3 *For any* Γ *the relation* $\sim^\Gamma$ *is transitive.*

Proof. An easy induction over the definition of $\sim^\Gamma$, using Lemma 7.2. ///

Lemma 7.4 *Let* Γ *be a valuation.*

1. $\Gamma \vdash M = N$ *iff there exists a bar* $\mathcal{B}$ *for* Γ *such that for every* $\Gamma' \in \mathcal{B}$*,* $\Gamma' \vdash M = N$*.*

2. *For any* n*, if* $\Gamma \vdash_n M = N$ *then there is a bar* $\mathcal{B}$ *for* Γ *such that for every consistent* $\Gamma' \in \mathcal{B}$

$$M \; (\xleftrightarrow{\Gamma' w} \cup \sim_n^{\Gamma'})^* \; N.$$

Proof.

1. The "only if" direction is trivial; take $\mathcal{B} = \{\Gamma_0\}$, where Γ_0 is any finite subset of Γ such that $\Gamma_0 \vdash_n M = N$. For the other direction: if $\Gamma \not\vdash_n M = N$, the ByCases rule allows us to successively extend Γ to valuations resolving arbitrary rigid terms, while maintaining the non-provability of $M = N$. This precludes the construction of an appropriate bar.

2. This is a routine proof by induction on the structure of the derivation of $\Gamma \vdash_n M = N$.

///

7.2 The relation between reduction and $\sim$

We have seen that $\xrightarrow{\Gamma w}$-reduction is Church-Rosser, but that it is not enough to capture provability. Lemma 7.4 suggests that we analyze the interaction between the relations $\xleftrightarrow{\Gamma w}$ and $\sim^\Gamma$. The natural property to hope for is that $\xrightarrow{\Gamma w}$ be Church-Rosser modulo $\sim^\Gamma$. This property, investigated by Huet in [Hue80], ensures that for a reduction relation $\longrightarrow$ and an equivalence relation $\sim$, if t_1 and t_2 are in the equivalence relation generated by $(\longrightarrow \cup \sim)$ then in fact there are s_1 and s_2 such that $t_1 \twoheadrightarrow s_1, t_2 \twoheadrightarrow s_2$, and $s_1 \sim s_2$.

Of course we are interested in the situation with the reduction relation being $\xrightarrow{\Gamma w}$ and the equivalence relation $\sim^\Gamma$, especially in light of Lemma 7.4.

There are well-known necessary and sufficient conditions (given in [Hue80]) for a reduction to be Church-Rosser modulo a relation; these generalize the classical tests for confluence and, if the reduction is strongly normalizing, local confluence. But these techniques are difficult to apply directly to the present calculus, essentially due to the presence of β-reduction.

For readers familiar with Huet's paper, we note that the second condition there in the definition of local confluence modulo an equivalence relation fails in the present situation. For example, we have that $(\lambda x.x)a \sim^\emptyset (\lambda x.(\mathsf{case}\ x\ \mathsf{in}_1\ \mathsf{in}_2))a$ since the two abstraction terms are provably equal. Huet's local confluence modulo $\sim^\emptyset$ would require, for example, that $(\lambda x.x)a$ and $(\mathsf{case}\ a\ \mathsf{in}_1\ \mathsf{in}_2)$ have reducts which are $\sim^\emptyset$-related. But this is easily seen to fail.

So indeed, it is false that $\xrightarrow{\Gamma w}$ is Church-Rosser modulo $\sim^\Gamma$ for arbitrary Γ. But when $M(\xleftrightarrow{\Gamma w} \cup \sim^\Gamma)^* N$ and Γ is sufficiently full for M and N, in the sense that it resolves M and N, then we can find a common reduct for M and N modulo $\sim^\Gamma$. This will be enough for our purposes.

Notation 7.5 Let $\mathcal{B}$ be a bar. Write

$$U \Downarrow^{\mathcal{B}} V$$

to mean that for every *consistent* $\Gamma \in \mathcal{B}$, $U_\Gamma \sim^\Gamma V_\Gamma$.

Definition 7.6 Let Γ be a finite valuation. $\mathcal{C}_\Gamma$ is a type-indexed family $\{\mathcal{C}_\Gamma^\sigma \mid \sigma \in \text{Types}\}$ of relations on closed terms defined as follows by induction on types:

1. At base types τ, $\mathcal{C}_\Gamma^\tau(M,N)$ if there is a bar $\mathcal{B}$ for Γ resolving M and N, such that $M \Downarrow^{\mathcal{B}} N$.
2. $\mathcal{C}_\Gamma^{\sigma_1 \times \sigma_2}(M,N)$ if for $i = 1,2$, $\mathcal{C}_\Gamma^{\sigma_i}(\mathsf{pr}_i M, \mathsf{pr}_i N)$,
3. $\mathcal{C}_\Gamma^{\sigma_1 + \sigma_2}(M,N)$ if there is a bar $\mathcal{B}$ for Γ resolving M and N, such that for every *consistent* $\Gamma' \in \mathcal{B}$, there is an index i and terms X and Y such that $M_{\Gamma'} \equiv \mathsf{in}_i X$, $N_{\Gamma'} \equiv \mathsf{in}_i Y$, and $\mathcal{C}_{\Gamma'}^{\sigma_i}(X,Y)$.
4. $\mathcal{C}_\Gamma^{\sigma \to \tau}(M,N)$ if for every valuation $\Gamma' \supseteq \Gamma$, $\mathcal{C}_{\Gamma'}^\sigma(A,B)$ implies $\mathcal{C}_{\Gamma'}^\tau(MA,NB)$.

If $\mathcal{B}$ is a bar for some valuation Γ, $\mathcal{C}_{\mathcal{B}}(M,N)$ asserts that $C_{\Gamma'}(M,N)$ for every $\Gamma' \in \mathcal{B}$.

The following are elementary properties of the logical relation.

Lemma 7.7 *Let M and N be closed, and σ be any type.*

1. *If Γ is a valuation, $\mathcal{B}_1 \preceq \mathcal{B}_2$ are bars for Γ, $\mathcal{B}_1$ is full for M and N and $M {\Downarrow}^{\mathcal{B}_1} N$ then $M {\Downarrow}^{\mathcal{B}_2} N$.*

2. *If $\mathcal{B}_1 \preceq \mathcal{B}_2$ are bars for finite valuation Γ and $\mathcal{C}_{\mathcal{B}_1}(M, N)$, then $\mathcal{C}_{\mathcal{B}_2}(M, N)$.*

3. *If $\mathcal{C}^{\sigma}_{\Gamma}(M, N)$ and $\Gamma \subseteq \Gamma^+$ then $\mathcal{C}^{\sigma}_{\Gamma^+}(M, N)$.*

4. *If $\mathcal{C}^{\sigma}_{\Gamma}(M, N)$ and $M \xleftrightarrow{\Gamma w} M'$ then $\mathcal{C}^{\sigma}_{\Gamma}(M', N)$.*

5. *$\mathcal{C}^{\sigma}_{\Gamma}$ is a partial equivalence relation on closed terms.*

6. *$\mathcal{C}^{\sigma}_{\Gamma}(M, N)$ iff there is a bar $\mathcal{B}$ for Γ satisfying$\mathcal{C}^{\sigma}_{\mathcal{B}}(M, N)$.*

Proof.

1. An easy consequence of the definitions.

2. Choose $\Gamma^+ \in \mathcal{B}_2$; by definition $\exists \Gamma \in \mathcal{B}_1$ such that $\Gamma \subseteq \Gamma^+$. We know $\mathcal{C}_{\Gamma}(M, N)$. $\mathcal{C}_{\Gamma^+}(M, N)$ is shown induction over types. At base types τ: let $\mathcal{B}$ be the bar witnessing $\mathcal{C}^{\tau}_{\Gamma}(M, N)$, and let $\Gamma \subseteq \Gamma^+$. Since $\mathcal{B}$ is also a bar for Γ^+ we simply observe that $M_{\Gamma^+} \equiv M_{\Gamma}$, $N_{\Gamma^+} \equiv N_{\Gamma}$, and the relation $\sim^{\Gamma}$ is contained in $\sim^{\Gamma^+}$.

   At sum types, we may again use the same bar as given by the hypothesis and use the induction hypothesis.

   At product types an easy application of the induction hypothesis suffices.

   At arrow types $\sigma = \sigma_1 \to \sigma_2$, let $\Gamma' \supseteq \Gamma$ and suppose $\mathcal{C}^{\sigma_1}_{\Gamma'}(A, B)$. We require $\mathcal{C}^{\sigma_2}_{\Gamma'}(MA, NB)$, but this follows directly from the induction hypothesis since $\mathcal{C}^{\sigma_2}_{\Gamma}(MA, NB)$.

3. Easy corollary of the previous part noting that $\{\Gamma\} \preceq \{\Gamma^+\}$.

4. An easy induction over types, using, at base and sum types, the fact that $\xrightarrow{\Gamma w}$ is Church-Rosser.

5. Symmetry follows from the symmetry of ${\Downarrow}^{\mathcal{B}}$.

   To see transitivity at base types: suppose $\mathcal{C}_{\Gamma}(M, N)$ and $\mathcal{C}_{\Gamma}(N, P)$ are witnessed by bars $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, that is, $M {\Downarrow}^{\mathcal{B}_1} N$ and $N {\Downarrow}^{\mathcal{B}_2} P$. By part (1), $M {\Downarrow}^{\mathcal{B}_1 \sqcup \mathcal{B}_2} N$ and $N {\Downarrow}^{\mathcal{B}_1 \sqcup \mathcal{B}_2} P$. Now use transitivity of ${\Downarrow}^{\mathcal{B}_1 \sqcup \mathcal{B}_2}$. The proof for sum types is similar; the case of product types is an easy application of induction.

   For an arrow type $\sigma_1 \to \sigma_2$: suppose $\mathcal{C}_{\Gamma}(M, N)$ and $\mathcal{C}_{\Gamma}(N, P)$. Choose $\Gamma' \supseteq \Gamma$ and suppose $\mathcal{C}_{\Gamma'}(A, B)$; we seek $\mathcal{C}_{\Gamma'}(MA, PB)$. Note that $\mathcal{C}_{\Gamma'}(A, A)$ by symmetry and transitivity of $\mathcal{C}^{\sigma_1}_{\Gamma'}$; so $\mathcal{C}_{\Gamma'}(MA, NA)$. Since $\mathcal{C}_{\Gamma'}(NA, PB)$ we can apply transitivity of $\mathcal{C}^{\sigma_2}_{\Gamma'}$.

6. For the left-to-right direction we may take the bar $\mathcal{B}$ to be $\{\Gamma\}$.

The converse proceeds by induction on types. At base types we have that for each $\Gamma' \in \mathcal{B}$ there is a bar $\mathcal{B}_{\Gamma'}$ which resolves M and N, and further for each $\Gamma'' \in \mathcal{B}_{\Gamma'}$, $M_{\Gamma''} \sim^{\Gamma''} N_{\Gamma''}$. Now just observe that $\bigcup\{\mathcal{B}_{\Gamma'} \mid \Gamma' \in \mathcal{B}\}$ is a bar for Γ which resolves M and N.

The argument for sum types is similar. For product types the result follows by an appeal to the inductive hypothesis.

At arrow types: given Γ and $\mathcal{B}$ such that $\mathcal{C}_{\Gamma'}(M,N)$ at each $\Gamma' \in \mathcal{B}$, choose $\Gamma^+ \supseteq \Gamma$ $\mathcal{C}_{\Gamma^+}$-related (A,B); we seek $\mathcal{C}_{\Gamma^+}(MA,NB)$. There are two cases: if there exists $\Gamma' \in \mathcal{B}$ with $\Gamma' \subseteq \Gamma^+$, then $\mathcal{C}_{\Gamma^+}(M,N)$ (using (1) above) so that $\mathcal{C}_{\Gamma^+}(MA,NB)$ immediately. Otherwise we consider the following bar for Γ^+ : $\mathcal{B}^+ = \mathcal{B} \cap \{\Delta \mid \Gamma^+ \subseteq \Delta.\}$ For each $\Delta' \in \mathcal{B}^+$ we have $\mathcal{C}_{\Delta'}(MA,NB)$; now use the induction hypothesis.

Lemma 7.8 *Let Γ be consistent. For each type σ,*

1_σ *If $\mathcal{C}^\sigma_\Gamma(M,N)$ then there is a bar $\mathcal{B}$ for Γ resolving M and N such that $M \Downarrow^{\mathcal{B}} N$.*

2_σ *$\mathcal{C}^\sigma_\Gamma(c,c)$ for constants $c \in \Sigma$.*

Proof. By induction on types. Note that the truth of $(1)_\sigma$ implies that if $\mathcal{C}^\sigma_\Gamma(M,N)$ then $\Gamma \vdash M = N$.

At base types. (1) holds by definition, and (2) follows from the fact that $c \sim^\Gamma c$ for all c.

At sum types. For (1) suppose $\mathcal{C}^{\sigma_1+\sigma_2}_\Gamma(M,N)$ and let $\mathcal{B}$ be the bar witnessing this. Build the collection $\mathcal{B}^*$ of bars as follows: if $\Gamma' \in \mathcal{B}$ is inconsistent, let $\mathcal{B}_{\Gamma'}$ be $\{\Gamma'\}$. Otherwise we have $M \xrightarrow{\Gamma' w}\!\!\!\!\!\rightarrow \mathsf{in}_i X$, $N \xrightarrow{\Gamma' w}\!\!\!\!\!\rightarrow \mathsf{in}_i Y$ and $\mathcal{C}^{\sigma_i}_{\Gamma'}(X,Y)$; let $\mathcal{B}_{\Gamma'}$ be the bar obtained by the induction hypothesis at $\mathcal{C}^{\sigma_i}_{\Gamma'}(X,Y)$. Then $\bigcup \mathcal{B}^*$ is a bar and satisfies our requirements.

For (2) we must show that $\mathcal{C}_\Gamma(c,c)$. If $c = \mathsf{in}_i(c_i) \in \Gamma$ then $c_\Gamma = \mathsf{in}_i(c_i)$, by induction hypothesis $c_i \Gamma c_i$, and it follows that $\{\Gamma\}$ is the desired bar. Otherwise, for $i = 1,2$ define Γ_i to be $\Gamma, c = \mathsf{in}_i c_i$. This forms a bar, and it is easy to check (using the induction hypothesis for the c_i) that it witnesses $\mathcal{C}_\Gamma(c,c)$.

At product types. For (1), suppose that $\mathcal{C}^{\sigma_1\times\sigma_2}_\Gamma(M,N)$. Then for $i = 1,2$, $\mathcal{C}^{\sigma_i}_\Gamma(\mathsf{pr}_i M, \mathsf{pr}_i N)$.

By induction hypothesis, there are bars $\mathcal{B}_i$ for Γ resolving M and N such that $\mathsf{pr}_i M \Downarrow^{\mathcal{B}_i} \mathsf{pr}_i N$. Let $\mathcal{B} = \mathcal{B}_1 \sqcup \mathcal{B}_2$; we know by Lemma 7.7 that $\mathsf{pr}_i M \Downarrow^{\mathcal{B}} \mathsf{pr}_i(N)$.

Let $\Gamma' \in \mathcal{B}$ be consistent. We may write $M_{\Gamma'} \equiv \langle M_1, M_2\rangle$ and $N_{\Gamma'} \equiv \langle N_1, N_2\rangle$. Since $(\mathsf{pr}_i(M))_\Gamma \equiv M_i$ and $(\mathsf{pr}_i(N))_\Gamma \equiv N_i$, and $M_i \sim^{\Gamma'} N_i$. Therefore $\langle M_1, M_2\rangle \sim^{\Gamma'} \langle N_1, N_2\rangle$, and thus $\mathcal{C}_{\Gamma'}(M,N)$.

For (2), to conclude $\mathcal{C}^{\sigma_1\times\sigma_2}_\Gamma(c,c)$ it suffices, by Lemma 7.7. 4, to show $\mathcal{C}^{\sigma_1\times\sigma_2}_\Gamma(\langle c_1,c_2\rangle, \langle c_1,c_2\rangle)$. For this it suffices to show $\mathcal{C}^{\sigma_i}_\Gamma(c_i,c_i)$ for $i = 1,2$. But these follow by induction.

At function types. For (1), suppose $\mathcal{C}_\Gamma(F,G)$. Construct a bar $\mathcal{B}$ for Γ resolving F and G. At arbitrary consistent $\Gamma' \in \mathcal{B}$ suppose $F_{\Gamma'} \equiv \lambda x.U$ and $G_{\Gamma'} \equiv \lambda x.V$; to conclude $F_{\Gamma'} \sim^{\Gamma'} G_{\Gamma'}$ it suffices to show that $\Gamma' \vdash U[c/x] = V[c/x]$ for a fresh constant c. But for such a c the induction hypothesis yields $\mathcal{C}_{\Gamma'}(c,c)$ and so $\mathcal{C}_{\Gamma'}((\lambda x.U)c, (\lambda x.V)c)$. This suffices, since $\Gamma' \vdash (\lambda x.U)c = (\lambda x.V)c)$. by induction and the observation at the start of the proof.

For (2) suppose $\vec{A}$ and $\vec{B}$ are such that $c\vec{A}$ and $c\vec{B}$ are of non-arrow type τ; by monotonicity we may take Δ to be a valuation such that for each i, $\mathcal{C}_\Delta(A_i, B_i)$. We seek to establish $\mathcal{C}^\tau_\Delta(c\vec{A}, c\vec{B})$.

If τ is a base type, apply the induction hypothesis to each $A_i \in \vec{A}$ and $B_i \in \vec{B}$ and combine the resulting bars using the $\sqcup$-construction we arrive at a bar $\mathcal{B}$ for Δ such that f for each i, $A_i \Downarrow^{\mathcal{B}} B_i$. We conclude that for any $\Delta' \in \mathcal{B}$, $(c\vec{A})_{\Delta'} \sim^{\Delta'} (c\vec{B})_{\Delta'}$ since no reductions occur at the root of the term. This establishes $\mathcal{C}^\tau_\Delta(c\vec{A}, c\vec{B})$.

If $\tau = \tau_1 \times \tau_2$, then by induction hypothesis $\mathcal{C}_\Delta(c_1\vec{A}, c_1\vec{B})$ and $\mathcal{C}_\Delta(c_2\vec{A}, c_2\vec{B})$. Noting that $\mathsf{pr}_i(c\vec{A}) \xrightarrow{w}\!\!\!\!\rightarrow c_i\vec{A}$ and $\mathsf{pr}_i(c\vec{B}) \xrightarrow{w}\!\!\!\!\rightarrow c_i\vec{B}$, and using Lemma 7.7(2), it follows that $\mathcal{C}_\Delta(\mathsf{pr}_i(c\vec{A}), \mathsf{pr}_i(c\vec{B}))$.

If $\tau = \tau_1 + \tau_2$, then let Δ_{ij} for $i,j = 1,2$, be obtained by adding to Δ the equations $c\vec{A} = \mathsf{in}_i(c_i\vec{A})$ if Δ does not already resolve $c\vec{A}$, and $c\vec{B} = \mathsf{in}_j(c_i\vec{B})$ if Δ does not resolve $c\vec{B}$. This defines a bar for Δ. When $i \neq j$ Δ_{ij} is inconsistent. To see this note that $A_i \Downarrow^{\Delta'} B_i$, for any $\Delta' \in \mathcal{B}$, and hence $\Delta' \vdash c\vec{A} = c\vec{B}$. By Lemma 7.4, $\Delta \vdash c\vec{A} = c\vec{B}$.

When $i = j$, $\mathcal{C}^\tau_{\Delta_{ij}}(\mathsf{in}_i(c_i), \mathsf{in}_j(c_j))$ by induction hypothesis. This then establishes $\mathcal{C}^\tau_\Delta(c\vec{A}, c\vec{B})$.

///

We have observed that $\mathcal{C}^\sigma_\Gamma$ is a partial equivalence relation in Lemma 7.7(3), but we can now prove reflexivity as well, using Lemma 7.8.

Lemma 7.9 $\mathcal{C}_\Gamma$ *is an equivalence relation on closed terms.*

Proof. We need only show reflexivity.

If θ_1 and θ_2 are substitutions, write $\mathcal{C}_\Gamma(\theta_1, \theta_2)$ if for all x in the domain of either substitution, $\mathcal{C}_\Gamma(\theta_1(x), \theta_2(x))$. Now define the relation $\mathcal{C^*}_\Gamma$ on open terms by: $\mathcal{C^*}_\Gamma(M, N)$ if whenever $\mathcal{C}_\Gamma(\theta_1, \theta_2)$ and $\theta_1 M$ and $\theta_2 N$ are closed, then $\mathcal{C}_\Gamma(\theta_1 M, \theta_2)$.

It suffices to prove that for all M, $\mathcal{C^*}_\Gamma(M, M)$; we do so by induction on M. When M is a variable this is immediate; when M is an application this is an easy application of the induction hypothesis. When M is $\lambda x.B$: let appropriate θ_1 and θ_2 be given; we may assume that x is not in the domain of either, so that we want to show $\mathcal{C}_\Gamma(\lambda x.\theta_1 B, \lambda x.\theta_2 B)$. So suppose $\mathcal{C}_\Gamma(A, A')$; we claim that $\mathcal{C}_\Gamma((\lambda x.\theta_1 B)A, (\lambda x.\theta_2 B)A')$. By Lemma 4 is suffices to show $\mathcal{C}_\Gamma(\theta_1 B)[A/x], (\theta_2 B)[A'/x])$. But these terms are of the form $\theta'_1 B$ and $\theta'_2 B$ respectively, with $\mathcal{C}_\Gamma(\theta'_1, \theta'_2)$.

It remains to show that for any constant c, $\mathcal{C}_\Gamma(c, c)$. For $c \in \Sigma$ this is Lemma 7.8. The remaining cases are pr_i, in_i, and case. Verification for the pr_i and in_i is straightforward. For case, choose $\vec{A}$, $\vec{B}$ and Δ just as in the proof of Lemma 7.8. Since $\mathcal{C}_\Delta(A_1, B_1)$ there is a bar $\mathcal{B}$ for Δ such that for consistent $\Delta' \in \mathcal{B}$, there are i, X, and Y with $A_i \xrightarrow{\Delta' w}\!\!\!\!\rightarrow in_i X$, $B_i \xrightarrow{\Delta' w}\!\!\!\!\rightarrow in_i X$, and $\mathcal{C}^{(}_{\Delta'} X, Y)$. It will suffice to show that $\mathcal{C}_{\Delta'}((\mathsf{case}\ A_1\ A_2\ A_3 \cdots A_n), (\mathsf{case}\ B_1\ B_2\ B_3 \cdots B_n)$ at these Δ'. Moreover it suffices to reduce and show $\mathcal{C}_{\Delta'}((A_i X \cdots A_n), (B_i Y \cdots B_n))$. But this follows from the fact that the relevant A_k and B_k are $\mathcal{C}_\Delta$-related.

///

Lemma 7.10 *If $M \overset{\Gamma w}{\longleftrightarrow} N$ then $\mathcal{C}_\Gamma(M,N)$.*

Proof. Use Lemma 7.7(2), the Church-Rosser property for $\overset{\Gamma w}{\longleftrightarrow}$, and reflexivity of $\mathcal{C}_\Gamma$. ///

Theorem 7.11 *Suppose Γ is a finite valuation and M and N are closed. If $\Gamma \vdash M = N$ there is a bar $\mathcal{B}$ for Γ resolving M and N such that $M \Downarrow^{\mathcal{B}} N$.*

Proof.

It suffices to show that if $\Gamma \vdash M = N$ then $\mathcal{C}_\Gamma(M,N)$.

We show simultaneously by induction on n:

1_n. For all Γ, $M \sim_n^\Gamma N$ implies $\mathcal{C}_\Gamma(M,N)$.

2_n. For all Γ, $\Gamma \vdash_n M = N$ implies $\mathcal{C}_\Gamma(M,N)$.

We first show that at each n, (1_n) implies (2_n). So suppose $\Gamma \vdash_n M = N$. By Lemma 7.4 there is a bar $\mathcal{B}$ for Γ such that for every $\Gamma' \in \mathcal{B}$

$$M \ (\overset{\Gamma' w}{\longleftrightarrow} \cup \sim^{\Gamma'})^* \ N.$$

Under (1_n), $\mathcal{C}_{\Gamma'}$ contains $\sim_n^\Gamma$. By Lemma 7.10. $\mathcal{C}_{\Gamma'}$ contains $\overset{\Gamma' w}{\longleftrightarrow}$. Now apply Lemma 7.9.

So at each n we address ourselves to (1_n) only, and will assume (2_k) for $k < n$. We use a sub-induction on the definition of $M \sim_n^\Gamma N$.

Note that if $M \equiv N$ then $\mathcal{C}_\Gamma(M,N)$ since $\mathcal{C}_\Gamma$ is reflexive — this already takes care of the case when $n = 0$.

If $M \sim_n^\Gamma N$ is derived via

$$\frac{F \sim_n^\Gamma F' \quad A \sim_n^\Gamma A'}{(FA) \sim_n^\Gamma (F'A')}$$

$\mathcal{C}_\Gamma(FA, F'A')$ by two easy applications of the induction hypothesis and the definition of logical relation.

Suppose

$$\frac{\Gamma \vdash_{n-1} U[c/x] = V[c/x]}{M \equiv \lambda x.U \sim_n^\Gamma \lambda x.V \equiv N} \quad c \text{ not occurring in } \Gamma.$$

Let Γ', P and Q be such that $\Gamma \subseteq \Gamma'$ and $\mathcal{C}_{\Gamma'}(P,Q)$; we seek $\mathcal{C}_{\Gamma'}(MP, NQ)$; it suffices to show $\mathcal{C}_{\Gamma'}(U[P/x], V[Q/x])$. We show that $\mathcal{C}_{\Gamma'}(U[P/x], V[P/x])$ and $\mathcal{C}_{\Gamma'}(V[P/x], V[Q/x])$ and invoke transitivity.

To see that $\mathcal{C}_{\Gamma'}(V[P/x], V[Q/x])$ holds, note that $\mathcal{C}_{\Gamma'}(\lambda x.V, \lambda x.V)$ and so $\mathcal{C}_{\Gamma'}((\lambda x.V)P, (\lambda x.V)Q)$, now reduce.

To see that $\mathcal{C}_{\Gamma'}(U[P/x], V[P/x])$: note that $\Gamma \vdash_{n-1} U[P/x] = V[P/x]$ since $c \notin \Gamma$; now apply the global induction hypothesis to conclude $\mathcal{C}_\Gamma(U[P/x], V[P/x])$ and by Lemma 7.7(1) $\mathcal{C}_{\Gamma'}(U[P/x], V[P/x])$ as well.

///

Recall that we refer to the constants among the in_i, $\langle\cdot,\cdot\rangle$, $*$, and the sets Σ and $\Sigma^\star$ as *passive constants*.

Lemma 7.12 *Let Γ be a consistent valuation, and suppose $(hA_1\cdots A_n)_\Gamma \sim^\Gamma (h'A'_1\cdots A'_{n'})$ where the two terms are of non-arrow type, and h and h' are passive. Furthermore suppose that Γ resolves these two terms. Then $h\equiv h'$, $n=n'$, and for $1\le i\le n$, $A_i \sim^\Gamma A'_i$.*

Proof. By induction on types. The lemma is clear when either of h or h' is in $\{\langle\cdot,\cdot\rangle,\mathsf{in}_i,*\}$. Let $h,h'\in\Sigma^\star$. Suppose the terms above have base type. Then

$$(hA_1\cdots A_n)_\Gamma \equiv (h(A_1)_\Gamma\cdots(A_n)_\Gamma)$$

and similarly for $(h'A'_1\cdots A'n')$. The result follows immediately from the definition from $\sim^\Gamma$.

If the terms have sum types, then since Γ resolves the terms in question we have

$$(h\vec{A})_\Gamma \equiv \mathsf{in}_i((h_i\vec{A})_\Gamma)$$

$$(h'\vec{A}')_\Gamma \equiv \mathsf{in}_{i'}((h'_{i'}\vec{A}')_\Gamma)$$

Note that by the definition of $\sim$, $(h_i\vec{A})_\Gamma \sim^\Gamma (h'_{i'}\vec{A}')_\Gamma$. By induction hypothesis, $i=i'$ and $h_i\equiv h'_{i'}$, and so $h\equiv h'$.

The case for product types is similar.

///

Theorem 7.13 *Let Γ be a consistent valuation. If $\Gamma\vdash hA_1\cdots A_n = h'A'_1\cdots A'_{n'}$ with h and h' passive constants, then $h\equiv h'$, $n=n'$, and for $1\le i\le n$, $\Gamma\vdash A_i=A'_i$*

Proof. If the terms are not of arrow type, we may choose fresh constants from Σ that drive the terms to non-arrow type. So without loss of generality assume that the equation is not of arrow type.

Suppose $\Gamma\vdash hA_1\cdots A_n = h'A'_1\cdots A'_{n'}$ and let $\mathcal{B}$ be a bar as provided by Theorem 7.11. For each consistent $\Gamma'\in\mathcal{B}$,

$$(hA_1\cdots A_n)'_\Gamma \sim^{\Gamma'} (h'A'_1\cdots A'n')'_\Gamma.$$

Note that since Γ is consistent, it follows from Lemma 7.4 that any bar for Γ contains a consistent valuation, say Γ'. So Lemma 7.12 can be applied to yield that $h\equiv h'$ and $n=n'$. Now the fact that $A_i \sim^{\Gamma'} A'_{i'}$, and therefore $\Gamma'\vdash A_i=A'_i$ for all consistent Γ' in $\mathcal{B}$, is enough by Lemma 7.4(1) to yield $\Gamma\vdash A_i=A'_i$. ///

It has taken a lot of work to establish the above decomposition property, but it is crucial to the construction in the proof of Proposition 8.1 below, which is in turn the key to ensuring extensionality in the term model we construct for the completeness theorem.

Some observations concerning the decomposition property: it is surprising at first glance that the result holds not just for the case $\emptyset \vdash hA_1 \cdots A_n = h'A'_1 \cdots A'n'$, but under an arbitrary valuation. This would not be true if we were to take an arbitrary set of hypotheses instead of a valuation. This is evidence that the use of valuation-equations as hypotheses is closer in spirit to reasoning in a pure calculus than in one with non-logical axioms.

In the absence of hypotheses, the decomposition property is easy to demonstrate when (in contrast to the present situation) provability is completely captured by a confluent rewriting system. But one should not necessarily expect it for arbitrary typed lambda calculi. For example, consider the full type hierarchy over a two element base type, ι, and only function types. Let f and x be variables of type $\iota \to \iota$ and ι respectively. The terms $(f\ (f\ (f\ x)))$ and $(f\ x)$ are equal in this model, while $(f\ (f\ x))$ and x are not. For another example, consider the continuous type hierarchy over a flat CPO; extend the simply-typed lambda calculus with a constant Ω. Let Ω denote the least element in the flat CPO. Then the equation $(f\ (f\ \Omega)) = (f\ \Omega)$ is true in the model but $(f\ \Omega) = \Omega$ is not. The decomposition property also fails if we consider the equational theory of the call-by-value lambda calculus interpreted over the strict continuous hierarchy over a flat CPO, as well as the lazy lambda calculus interpreted over the type hierarchy over a flat CPO with arrow types interpreted by lifted continuous function spaces.

8 Completeness

We recall the discussion at the end of Section 4, in which we noted that working with infinite sets of equations causes problems in trying to ensure extensionality for the corresponding term structure.

The solution is to encode negation, in the following sense.

Notation

Write $\Gamma \Vdash M \neq N$ to mean that for every consistent $\Gamma' \supseteq \Gamma$, $\Gamma' \nvdash M = N$.

Proposition 8.1 *Let Γ be a finite valuation such that $\Gamma \nvdash M = N$. Then there exists a consistent finite valuation Γ^+ extending Γ such that $\Gamma^+ \Vdash M \neq N$.*

Proof.

Since $\Longrightarrow$ is terminating, so its multiset extension $\Longrightarrow^m$ [DM79]. We will prove the theorem by Noetherian induction (well-founded induction over the converse of a terminating relation, [Coh81]); in this case over $\Longrightarrow^m$ applied to the pair (M, N).

- If at least one of M and N is $\longrightarrow \Gamma w$-reducible, then $\Gamma \not\vdash M_\Gamma = N_\Gamma$ and the multiset $\{M_\Gamma, N_\Gamma\}$ is $\Longrightarrow^m$-related to $\{M, N\}$; invoke the induction hypothesis.

- Otherwise, if the type of M and N is an arrow type, then $M \equiv \lambda x.U$ and $N \equiv \lambda x.V$; choose a fresh constant c, note that $\Gamma \not\vdash U[c/x] = V[c/x]$ and that these terms are obtained by Add-Argument, so induction applies.

- If $M \equiv c\vec{A}$ and $N \equiv c\vec{B}$, then there must be an i such that $\Gamma \not\vdash A_i = B_i$; these are Select-reducts of M and N, and hence $\{A_i, B_i\}$ is lower in the multiset ordering. By induction hypothesis there is $\Gamma^+ \supseteq \Gamma$ such that $\Gamma^+ \Vdash A_i \neq B_i$. By Theorem 7.13 $\Gamma^+ \Vdash M \neq N$.

- If $M \equiv c\vec{A}$ and $N \equiv d\vec{B}$, with c, d passive then again by Theorem 7.13, we conclude that in fact $\Gamma \Vdash M \neq N$ (no consistent valuation can equate two terms with different heads).

- The only other possibility is that M (say) is of the form (`case` $R\ F\ G$) with R rigid — that is to say, M is not resolved. Let $\mathcal{B}$ be a bar resolving M and N; by Lemma 7.4 we may select $\Gamma' \in \mathcal{B}$, $\Gamma \subseteq \Gamma'$, $\Gamma' \not\vdash M = N$ (for the latter see Lemma 7.4). Note that $M \xrightarrow{\Gamma'}\!\!\!\!\rightarrow M_{\Gamma'}$ but not $M \equiv M_{\Gamma'}$, so the induction hypothesis applies.

///

Proposition 8.2 *Let Γ be a finite valuation. If $\Gamma \not\vdash M = N$ then there is a countable d.p. model for Γ in which $M = N$ fails.*

Proof. We will construct a consistent valuation Γ_∞ such that

- Γ_∞ is full;
- for any F and G, if $\Gamma_\infty \not\vdash F = G : \sigma \to \tau$ then there is a term A such that $\Gamma_\infty \not\vdash FA = GA : \tau$; and
- $\Gamma_\infty \not\vdash M = N$.

The closed term structure for Γ_∞ will be the desired model.

We build Γ_∞ as the union of a chain of valuations Γ_n. List all closed terms of sum-type as $Q_{2k+1}, k \in \omega$; and list all closed equations $F = G$ between terms of arrow-type as $e_{2k+2}, k \in \omega$. Define $\Gamma_0 \supseteq \Gamma$ as in Lemma 8.1 so that $\Gamma_0 \Vdash M \neq N$. Let Γ_{2k+1} extend Γ_{2k} and resolve Q_{2k+1}. To define Γ_{2k+2}, consider $e_{2k+2} \equiv F = G : \sigma \to \tau$ and consider two cases. If $\Gamma_{2k+1} \vdash F = G$ then set $\Gamma_{2k+2} \equiv \Gamma_{2k+1}$. Otherwise, choose $c \in \Sigma$ not occurring in Γ_{2k+1}, and, noting $\Gamma_{2k+1} \not\vdash Fc = Gc$ let Γ_{2k+2} be as in Lemma 8.1, extending Γ_{2k+1} and such that $\Gamma_{2k+2} \Vdash Fc \neq Gc$.

The set $\Gamma_\infty \equiv \bigcup\{\Gamma_n \mid n \in \omega\}$ is easily seen to satisfy the three conditions above. ///

Theorem 8.3 (Completeness) *For any M and N, $\vdash M = N$ iff Set $\models M = N$.*

Proof. By Proposition 4.4 and Proposition 8.2. ///

9 Conclusion

We have established a fundamental model-theoretic property of the simply-typed λ-calculus with coproducts; this can be thought of as laying a foundation for a model theory of the calculus. But many questions remain.

One would like a characterization of those classes $\mathcal{C}$ of models for which validity in $\mathcal{C}$ implies provability (say, in the proof system given in this paper). This seems to be difficult: as we saw above, the 1-section theorem, which settles this question for the simply-typed lambda calculus, fails dramatically in the presence of coproducts.

Similarly one can ask whether a finite model theorem holds, as for the classical case [Sta82]. We conjecture that it does, that is, if an equation holds in every full type hierarchy with a finite base set, then it holds in *Set*.

References

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Volume 103 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam, 1981. Revised edition, 1984.

[Coh81] P.M. Cohn. Universal Algebra. Reidel, Dordrecht, Holland, 2nd ed. 1981.

[Cub92] D. Čubrić. Embedding of a free cartesian closed category into the category of Sets. Manuscript, McGill University, 1992

[DK93] R. Di Cosmo, D. Kesner. A confluent reduction system for the extensional λ-calculus with pairs, sums, recursion and terminal object. *Proc. ICALP* 1993.

[Dou93] D. Dougherty. Some λ-calculi with categorical sums and products. *Proc. Fifth Intl. Conf. on Rewriting Techniques and Applications,* Springer-Verlag, LNCS 690, (1993) 135–151.

[DM79] N. Dershowitz, Z. Manna. Proving termination with multiset orderings. *Comm. ACM* **22**(8), (1979) 465–476.

[Fri75] H. Friedman. Equality between functionals. In Rohit Parikh, ed., *Logic Colloquium '73*, Lec. Notes in Math. **453**, 22–37. Springer-Verlag, 1975.

[Hue80] G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. In *JACM 27:4*, pp. 797-821, 1980.

[LS86] J. Lambek, P. Scott. *Introduction to Higher-order Categorical Logic.* Cambridge Studies in Advanced Mathematics 7. Cambridge University Press 1986.

[Mit90] J. C. Mitchell. Type systems for programming languages. In Jan van Leeuwen (ed.), *Handbook of Theoretical Computer Science,* vol B, 365–458. Elsevier 1990.

[MM91] J. C. Mitchell, E. Moggi. Kripke-style models for typed lambda calculus. *J. Pure and Applied Logic, 51*, pp 99-124, 1991.

[OS91] M. Okada, P. Scott. Rewriting theory for uniqueness conditions: coproducts. Talk presented First Montreal Workshop on Programming Language Theory, April 1991.

[Plo80] G. D. Plotkin. Lambda definability in the full type hierarchy, in: P. Seldin and R. Hindley, eds., *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 363-373. Academic Press, New York, 1980.

[Sta82] R. Statman. Completeness, invariance, and lambda-definability. *J. Symbolic Logic,* vol 47, 17–26, 1982.

A Appendix: Proofs of Lemmas in Section 5

In the following the sign =, unless indicated otherwise, will mean provability in theory ABC.

Congruence with respect to application

Lemma A.1 *Define $f(\Upsilon)(i) \triangleq \lambda x_i.\lambda y.\Upsilon(i)\ x_i$.For terms M and P, and hypotheses list Δ, if the terms in the following equation type check then the equation holds.*

$$(\widehat{\Delta}_{f(\Upsilon_1)}[M]\ \widehat{\Delta}_{\Upsilon_2}[P]) = \widehat{\Delta}_{\Upsilon_1}[M\ P]$$

Note that the type of the bound variable y in the definition of f must be the same as the type of P in Δ.

Proof. The following equation is provable from empty hypotheses in ABC.

$$\mathsf{case}\ t\ (\lambda a.M)\ (\lambda b.N)(\mathsf{case}\ t\ (\lambda a.M')\ (\lambda b.N')) = \mathsf{case}\ t\ (\lambda a.M\ M')\ (\lambda b.N\ N')$$

The lemma follows by induction on the length of Δ. ///

Lemma A.2 *If for every Υ $\widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[N]$ and for every Υ $\widehat{\Delta}_\Upsilon[P] = \widehat{\Delta}_\Upsilon[Q]$ then for every Υ $\widehat{\Delta}_\Upsilon[M\ P] = \widehat{\Delta}_\Upsilon[N\ Q]$.*

Proof. By hypothesis, for every Υ_1 and Υ_2,

$$\begin{array}{lcl} \widehat{\Delta}_{f(\Upsilon_1)}[M] & = & \widehat{\Delta}_{f(\Upsilon_1)}[N] \\ \widehat{\Delta}_{\Upsilon_2}[P] & = & \widehat{\Delta}_{\Upsilon_2}[Q] \end{array}$$

Rewriting both terms in this equation using the equation in the previous lemma gives us the desired equation. ///

The ξ-rule

For $\forall i.\ x \neq x_i$ with x not free in U_i, define $g_x(\Upsilon)(i) \triangleq \lambda x_i.\Upsilon(1)\ x_i\ x$.

Lemma A.3 $\lambda x.\widehat{\Delta}_{g_x(\Upsilon)}[M] = \widehat{\Delta}_\Upsilon[\lambda x.M]$, *where x is neither free nor bound in $\widehat{\Delta}_\Upsilon[]$.*

Proof. If $\Delta \equiv []$ then $\widehat{\Delta}_{g_x(\Upsilon)}[M] \equiv M \equiv \widehat{\Delta}_\Upsilon[M]$; the claim follows trivially.
If $\Delta \equiv (t_1 = \mathsf{in}_1(a_1)) :: \Delta^-$, then

$$\begin{array}{rcll}
\lambda x.\widehat{\Delta}_{g_x(\Upsilon)}[M] & \equiv & \lambda x.\mathsf{case}\ t_1\ (\lambda a_1.\widehat{\Delta^-}_{g_x(\Upsilon^-)}[M])\ (\lambda x_1.\Upsilon(1)\ x_i\ x) & \\
& = & \mathsf{case}\ t_1\ (\lambda a_1.\lambda x.\widehat{\Delta^-}_{g_x(\Upsilon^-)}[M])\ (\lambda x_1.\lambda x.\Upsilon(1)\ x_i\ x) & \text{since } x \notin fv(t_1) \\
& = & \mathsf{case}\ t_1\ (\lambda a_1.\widehat{\Delta^-}_{\Upsilon^-}[\lambda x.M])\ (\lambda x_1.\lambda x.\Upsilon(1)\ x_i\ x) & \text{by induction} \\
& = & \mathsf{case}\ t_1\ (\lambda a_1.\widehat{\Delta^-}_{\Upsilon^-}[\lambda x.M])\ (\lambda x_1.\Upsilon(1)\ x_i) & \\
& = & \mathsf{case}\ t_1\ (\lambda a_1.\widehat{\Delta^-}_{\Upsilon^-}[\lambda x.M])\ (\Upsilon(1)) & \\
& \equiv & \widehat{\Delta}_\Upsilon[\lambda x.M] &
\end{array}$$

///

Lemma A.4 *If for every Υ, $\widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[N]$ then for every Υ, $\widehat{\Delta}_\Upsilon[\lambda x.M^x_c] = \widehat{\Delta}_\Upsilon[N^x_c]$, provided $c, x \notin \Delta$ and x is not free in M and N.*

Proof. Applying the ξ-rule to the hypothesis

$$\lambda x.\widehat{\Delta}_{g_x(\Upsilon)}[M^x_c] = \lambda x.\widehat{\Delta}_{g_x(\Upsilon)}[N^x_c]$$

Clearly x does not appear in $\widehat{\Delta}_\Upsilon[]$. Applying the previous lemma to the two terms in this equation, the claim follows. ///

The ByCases rule

Lemma A.5 *Let $\Delta^1 = \Delta.(t = \mathsf{in}_1(a))$ and $\Delta^2 = \Delta.(t = \mathsf{in}_2(b))$. If $\forall \Upsilon_1.\widehat{\Delta^1}_{\Upsilon_1}[M] = \widehat{\Delta^1}_{\Upsilon_1}[N]$ and $\forall \Upsilon_2.\widehat{\Delta^2}_{\Upsilon_2}[M] = \widehat{\Delta^2}_{\Upsilon_2}[N]$, and $a, b \notin M, N$ then $\forall \Upsilon\ \widehat{\Delta}_\Upsilon[M] = \widehat{\Delta}_\Upsilon[N]$.*

Proof. Let t have type $\tau_1+\tau_2$ in Δ. For any given Υ define $\Upsilon_1 = \Upsilon\cdot\lambda y.M$ and $\Upsilon_2 = \Upsilon\cdot\lambda x.N$, where $x : \tau_1$ and $y : \tau_2$ are free in neither M nor N. Recall that

$$\begin{array}{rcl}
\widehat{\Delta^1}_{\Upsilon_1}[] & \equiv & \widehat{\Delta}_\Upsilon[\mathsf{case}\ t\ (\lambda a.[])\ (\lambda x.M)] \\
\widehat{\Delta^2}_{\Upsilon_1}[] & \equiv & \widehat{\Delta}_\Upsilon[\mathsf{case}\ t\ (\lambda x.N)\ (\lambda b.[])]
\end{array}$$

Note that

$$
\begin{array}{rcl}
\widehat{\Delta}_{\Upsilon}[M] & = & \widehat{\Delta}_{\Upsilon}[\mathsf{case}\ t\ (\lambda a.M)\ (\lambda y.M)] \\
& = & \widehat{\Delta^1}_{\Upsilon_1}[M] \\
& = & \widehat{\Delta^1}_{\Upsilon_1}[N] \\
& = & \widehat{\Delta}_{\Upsilon}[\mathsf{case}\ t\ (\lambda a.N)\ (\lambda y.M)] \\
& = & \widehat{\Delta^2}_{\Upsilon_2}[M] \\
& = & \widehat{\Delta^2}_{\Upsilon_2}[N] \\
& = & \widehat{\Delta}_{\Upsilon}[\mathsf{case}\ t\ (\lambda x.N)\ (\lambda b.N)] \\
& = & \widehat{\Delta}_{\Upsilon}[N]
\end{array}
$$

In the first and last steps above we use the facts that none of a, b, x, or y are free in Mor in N. ///

Using Hypotheses

Lemma A.6 *Let* $\Delta = (t_1 = \mathsf{in}_{\delta_1}(a_1)), \ldots, (t_j = \mathsf{in}_{\delta_j}(a_j)), \underbrace{\ldots, (t_n = \mathsf{in}_{\delta_n}(a_n))}_{\Delta^0}$, *where* $\delta_j = 1$.

Then,

$$\widehat{\Delta}_{\Upsilon}[C[t_j]] = \widehat{\Delta}_{\Upsilon}[C[\mathsf{in}_1(a_j)]]$$

Proof. For ease of notation assume $\delta_j = 1$. Define context $D[]$ and sequence Υ_0 so that

$$\widehat{\Delta}_{\Upsilon}[] \ \equiv\ D[\mathsf{case}\ t_j\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[])\ (\Upsilon(j))]$$

So,

$$
\begin{array}{rcl}
\widehat{\Delta}_{\Upsilon}[C[t_j]] & = & D[\mathsf{case}\ t_j\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[C[t_j]])\ (\Upsilon(j))] \\
& = & D[\mathsf{case}\ t_j\ (X)\ (Y)] \\
& = & D[\mathsf{case}\ t_j\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[C[\mathsf{in}_1(a_j)]])\ (\lambda b'_j.\Upsilon(j)\ b'_j)] \\
& = & D[\mathsf{case}\ t_j\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[C[\mathsf{in}_1(a_j)]])\ (\Upsilon(j))] \\
& = & \widehat{\Delta}_{\Upsilon}[C[\mathsf{in}_1(a_j)]]
\end{array}
$$

where in the second line

$$
\begin{array}{rcl}
X & \equiv & \lambda a'_j.\mathsf{case}\ \mathsf{in}_1(a'_j)\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[C[\mathsf{in}_1(a'_j)]])\ (\Upsilon(j)) \\
Y & \equiv & \lambda b'_j.\mathsf{case}\ \mathsf{in}_2(b'_j)\ (\lambda a_j.\widehat{\Delta^0}_{\Upsilon_0}[C[\mathsf{in}_2(b'_j)]])\ (\Upsilon(j))
\end{array}
$$

where the variable b'_j in the term Y is fresh. ///

B Appendix: Strong Normalization of $\Longrightarrow$

This section contains the proof of strong normalization for $\Longrightarrow$ restricted to closed terms. The restriction to closed terms only comes into play in the final theorem; up to that point the words "terms" will mean terms possibly containing free variables.

Notation B.1 Write $\overset{-}{\Longrightarrow}$ for the reduction relation determined by $\Longrightarrow$ *without* the rules for Add-argument and Select. Note that these are precisely the rules which preserve types, and so can be applied to any subterm of a term

Write $\mathcal{SN}$ for the set of terms which are strongly normalizing with respect to $\Longrightarrow$-reduction.

Some preliminary observations about $\mathcal{SN}$ will be convenient:

Lemma B.2

1. *If* $\mathcal{SN}(P)$ *then* $\mathcal{SN}(\mathsf{in}_i P)$.
2. *If* $\mathcal{SN}(P_1)$ *and* $\mathcal{SN}(P_2)$, *then* $\mathcal{SN}(\langle P_1, P_2 \rangle)$.
3. *If* $\mathcal{SN}(B[c/x])$ *for all* $c \in \Sigma$ *then* $\mathcal{SN}(\lambda x.B)$.

Proof.

1. Any reduction sequence out of $\mathsf{in}_i P$ looks like

   $$\mathsf{in}_i P \overset{-}{\Longrightarrow}\!\!\!\succ \mathsf{in}_i P' \overset{-}{\Longrightarrow}\!\!\!\succ \cdots$$

   where $P \overset{-}{\Longrightarrow}\!\!\!\succ P'$, or perhaps

   $$\mathsf{in}_i P \overset{-}{\Longrightarrow}\!\!\!\succ \mathsf{in}_i P' \Longrightarrow\!\!\!\succ P' \Longrightarrow\!\!\!\succ \cdots$$

   In each case the reduction is finite by hypothesis on P.

2. The proof is similar to the previous case.

3. Any reduction sequence out of $\lambda x.B$ looks like

   $$\lambda x.B \overset{-}{\Longrightarrow}\!\!\!\succ \lambda x.B' \overset{-}{\Longrightarrow}\!\!\!\succ \cdots$$

   where $B \overset{-}{\Longrightarrow}\!\!\!\succ B'$, or perhaps

   $$\lambda x.B \overset{-}{\Longrightarrow}\!\!\!\succ \lambda x.B' \Longrightarrow\!\!\!\succ B'[c/x] \Longrightarrow\!\!\!\succ \cdots$$

   For the sake of contradiction suppose such a reduction to be infinite. In the first case we have an infinite $\overset{-}{\Longrightarrow}$ reduction out of B; letting c be any constant yields an infinite reduction out of $B[c/x]$. In the second case we may rearrange the reduction to do the Add-argument step first and again obtain an infinite reduction out of $B[c/x]$. This is a contradiction in each case.

///

Definition B.3 The type-indexed family of relations $\mathcal{S}^\sigma \subseteq \Lambda^\sigma_\Sigma$ is defined by induction on types as follows:

1. For $\sigma \in \{\iota, \mathbf{1}\}$, $\mathcal{S}^\sigma = \mathcal{SN}^\sigma$.
2. $\mathcal{S}^{\sigma_1+\sigma_2}(M)$ iff $\mathcal{SN}^{\sigma_1+\sigma_2}(M)$ and furthermore if $M \stackrel{-}{\Longrightarrow\!\!\!\!\succ} \mathsf{in}_i(N)$ then $\mathcal{S}^{\sigma_i}(N)$ $(i = 1, 2)$.
3. $\mathcal{S}^{\sigma_1\times\sigma_2}(M)$ iff $\mathcal{SN}^{\sigma_1\times\sigma_2}(M)$ and furthermore if $M \stackrel{-}{\Longrightarrow\!\!\!\!\succ} \langle M_1, M_2\rangle$ then $\mathcal{S}^{\sigma_1}(M_1)$ and $\mathcal{S}^{\sigma_2}(M_2)$.
4. $\mathcal{S}^{\sigma_1\to\sigma_2}(M)$ iff for every $N\colon \sigma_1$, $\mathcal{S}^{\sigma_1}(N) implies \mathcal{S}^{\sigma_2}(M\ N)$

Lemma B.4 *If $\mathcal{S}^\sigma(M)$ and $M \stackrel{-}{\Longrightarrow} M'$ then $\mathcal{S}^\sigma(M')$.*

Proof. By induction over types; the statement is clear at non-arrow types.

When $\sigma = \sigma_1 \to \sigma_2$. Let N be given such that $\mathcal{S}^{\sigma_1}(N)$; we seek to establish $\mathcal{S}^\sigma(M')$. But $\mathcal{S}^{\sigma_2}(MN)$, and $MN \stackrel{-}{\Longrightarrow} M'N$. By induction hypothesis $\mathcal{S}^{\sigma_2}(M'N)$.

///

Lemma B.5

1. *If $\mathcal{S}(P)$ and $\mathcal{SN}(P)$ then $\mathcal{S}(\mathsf{in}_i P)$.*
2. *If $\mathcal{S}(P_1), \mathcal{S}(P_2), \mathcal{SN}(P_1)$ and $\mathcal{SN}(P_2)$, then $\mathcal{S}(\langle P_1, P_2\rangle)$.*

Proof. In each case strong normalization follows from Lemma B.2. The second condition for membership in $\mathcal{S}$ follows similar lines, using the fact that $\mathcal{S}$ is closed under $\stackrel{-}{\Longrightarrow}$. ///

Recall that an *introduction* term is a term of the form $*$, $\mathsf{in}_i(M)$, $\langle M_1, M_2\rangle$, or $\lambda x.M$.

Lemma B.6 *Let M be a non-introduction term of non-arrow-type τ. If every one-step $\Longrightarrow$-reduct of M is strongly normalizing and in $\mathcal{S}$ then $\mathcal{S}(M)$.*

Proof. M is strongly normalizing if its one-step reducts are; the second condition on membership in $\mathcal{S}$ is clear since M is not an introduction. ///

Lemma B.7 *1_σ) $\mathcal{S}^\sigma \subseteq \mathcal{SN}^\sigma$.*

2_σ) $\Sigma^\sigma \subseteq \mathcal{S}^\sigma(f)$.

Proof. We prove these jointly by induction on the length of the type σ.

1_σ. If σ is a base type, a sum type or a product type then (1) holds by definition.

Suppose $\sigma = \sigma_1 \to \sigma_2$, and $\mathcal{S}^\sigma(F)$. Suppose, by way of contradiction, an infinite sequence

$$F \equiv F_0 \Longrightarrow F_1 \Longrightarrow F_2 \cdots .$$

Note that for any $d \in \Sigma^{\sigma_1}$, $\mathcal{S}^{\sigma_1}(d)$ by induction hypotheses (2_{σ_1}), and so $\mathcal{S}^{\sigma_2}(Fd)$. Then by induction hypothesis (1_{σ_2}), $\mathcal{SN}^{\sigma_1}(Fd)$. Now there are two cases to consider.

- The reduction sequence contains no Select, Add argument or η-expansion step applied to the root. Pick $d \in \Sigma^\sigma$. Then, $Fd \equiv F_0 d \Longrightarrow F_1 d \Longrightarrow F_2 d \ldots$ is an infinite reduction sequence, a contradiction.
- Otherwise let n be the smallest number such that $F_n \Longrightarrow F_{n+1}$ is a Select, Add-argument or η-expansion step.
  * If $F_n \Longrightarrow F_{n+1}$ is a Select step, i.e., $F_n = h\vec{A}$ and $F_{n+1} \equiv A_i$ then pick $d \in \Sigma^\sigma$, and note that the infinite reduction sequence $Fd \Longrightarrow F_1 d \Longrightarrow \ldots F_n d \Longrightarrow F_{n+1} \ldots$ contradicts $\mathcal{SN}^{\sigma_2}(Fd)$.
  * If $F_n \Longrightarrow F_{n+1} \equiv \lambda x.F_n x$ is an η-expansion, then since $F \overset{=}{\Longrightarrow\!\!\!\!\!\!\rightarrow} F_n$, F_n is in $\mathcal{S}$, so that for each d, $F_n d \in \mathcal{S}$, so the hypothesis of Lemma B.2 (part 3) is satisfied for F_n. This contradicts the assumption that the above reduction is infinite.
  * If $F_n \Longrightarrow F_{n+1}$ is an Add-argument step *ie.* $F_n = \lambda x.P$, and $F_{n+1} = P^c_x$, for some $c \in \Sigma^{\sigma_1}$, then $Fc \Longrightarrow F_1 d \Longrightarrow \ldots F_n c \xrightarrow{\beta} F_{n+1} \Longrightarrow \ldots$ is an infinite reduction sequence, again a contradiction.

2_σ) If $f{:}\,\sigma$ is of base type then f has no $\Longrightarrow$-reduct, hence is trivially $\mathcal{SN}$, and hence $\mathcal{S}(f)$.

For $f{:}\,\sigma_1 \times \sigma_2$ we invoke Lemma B.6 and check the one-step $\Longrightarrow$-reducts of f. The only such reduct of f is $\langle f_1, f_2 \rangle$, and by induction hypothesis (2_{σ_i}), $\mathcal{S}^{\sigma_i}(f_i)$; by induction hypothesis (1_{σ_i}) $\mathcal{SN}^{\sigma_i}(f_i)$. We now use Lemma B.5.

For $f{:}\,\sigma_1 + \sigma_2$ the argument is similar: note that the only one-step $\Longrightarrow$-reducts of f are of the form $\mathsf{in}_i(f_i)$.

If $\sigma = \sigma_0 \to \ldots \to \sigma_n \to \tau$, for non-arrow type τ, choose $\vec{M}$ satisfying $\mathcal{S}^{\sigma_i}(M_i)$. By induction hypothesis 1_{σ_i}), $\mathcal{SN}^{\sigma_i}(M_i)$. We show, by noetherian induction over $\Longrightarrow$-reduction over the multiset of the M_i, that all one step reducts of $f\vec{M}$ are in $\mathcal{SN}$ and in $\mathcal{S}$. Lemma B.6 will then yield $f\vec{M} \in \mathcal{S}$ as desired. The one-step reducts of $f\vec{M}$ are

- M_i, which is in $\mathcal{S}$ by assumption, and are in $\mathcal{SN}$ by induction hypothesis (1_{σ_i}).

- $fM_1 \ldots M_{i-1}M'_iM_i \ldots M_n$, where $M_i \overset{-}{\Longrightarrow} M'_i$. Apply induction.
- $\mathsf{in}_i(f_i\vec{M})$, if $\tau = \tau_1 + \tau_2$. But the type of f_i is shorter than σ, hence $\mathcal{S}(f_i)$, therefore $\mathcal{S}(f_i\vec{M})$, so by (1_τ) $\mathcal{SN}^{\tau_i}(f_i\vec{M})$, and by Lemma B.5 $\mathcal{S}(\mathsf{in}_i(f_i\vec{M}))$. Now use Lemma B.5
- $\langle f_1\vec{M}, f_2\vec{M}\rangle$, which is treated similar to the previous case.

///

Observe as a corollary to part (1) of the previous Lemma, that the "$\mathcal{SN}$" hypothesis in Lemma B.5 is now redundant.

Lemma B.8 *All constants are in $\mathcal{S}$.*

Proof. This has already been shown for constants in Σ. For in_i and $\langle .,. \rangle$ we invoke Lemma B.5. For $*$, we need only observe that it is irreducible, hence $\mathcal{SN}$. The constants pr_i and case are treated similarly: we outline the argument for case.

Suppose $Q, F_1, F_2, X_1, \ldots X_n, (n \geq 0)$ are in $\mathcal{S}$ and that $M \equiv (\mathsf{case}\ Q\ (F_1)\ (F_2))\vec{X}$ has non-arrow type; we wish to see that $M \in \mathcal{S}$. As in the previous Lemma, we argue by noetherian induction over $\Longrightarrow$ that the one-step reducts of M are in $\mathcal{S}$ (but now strong normalization comes for free). The only interesting case is when $Q \equiv \mathsf{in}_iX$ and a root-reduction occurs:

$$M \equiv (\mathsf{case}\ \mathsf{in}_iX\ (F_1)\ (F_2))\vec{X} \Longrightarrow F_iX \equiv M'$$

But certainly $\mathsf{in}_iX \in \mathcal{S}$ implies $X \in \mathcal{S}$ and so $M' \in \mathcal{S}$. ///

Lemma B.9 *For types σ_1 and σ_2, and term M assume that $\mathcal{S}(M)$, and that for every $N \in \mathcal{S}$, $\mathcal{S}(M[N/x])$ holds. Then $\mathcal{S}^{\sigma_1 \to \sigma_2}(\lambda x.M)$.*

Proof. Let $\sigma_2 = \tau_0 \to \ldots \tau_k$, where τ_k is not an arrow type. It suffices to show that for any $M_0, M_1, \ldots, M_k$ such that $\mathcal{S}^{\tau_i}(M_i)$ for each i, $\mathcal{S}^{\tau_k}((\lambda x.M)\ M_0 \ldots M_k)$. The form of the argument is by now familiar. We examine the one-step reducts, and note that the only situation which does not submit to the induction hypothesis is that of a head-β reduction. But such a reduction yields $(M[M_0/x])M_1 \cdots M_k$. But this is in $\mathcal{S}$ by our hypothesis on M. ///

Theorem B.10 *The relation $\Longrightarrow$ restricted to closed terms is strongly normalizing.*

Proof. For any term M, we will prove the following claim by induction over its structure.

> For any θ such that $M\theta$ is closed and such that for every x in the domain of $\theta, \mathcal{S}(\theta(x))$, $\mathcal{S}(M\theta)$.

If M is a variable, this is immediate. The cases where M is a constant is Lemma B.8. If M is an application $(P\ Q)$, then by induction hypothesis $\mathcal{S}(P\theta)$ and $\mathcal{S}(Q\theta)$, whence $\mathcal{S}((P\ Q)\theta)$. If $M \equiv \lambda x.P$ then by the previous lemma it suffices to show that for any Q satisfying $\mathcal{S}(Q)$ we have $\mathcal{S}(P\theta_x^Q)$. This is obvious by induction hypothesis applied to P. This completes the proof of the claim.

Now if M is any closed term, take θ to be the empty substitution; by the above claim $\mathcal{S}(M)$. By Lemma B.7(1) $\mathcal{SN}(M)$. ///

Theorem B.11 *The reduction* $\Longrightarrow$ *is strongly normalizing.*

Proof. Any infinite reduction sequence from an open term trivially "lifts" to an infinite reduction sequence from the closed term obtained by replacing each free variable by a constant throughout the sequence. ///