Normalization by evaluation for typed lambda calculus with coproducts

T. Altenkirch,* P. Dybjer,† M. Hofmann,‡ P. Scott§

Abstract

We solve the decision problem for simply typed lambda calculus with strong binary sums, equivalently the word problem for free cartesian closed categories with binary coproducts. Our method is based on the semantical technique known as "normalization by evaluation" and involves inverting the interpretation of the syntax into a suitable sheaf model and from this extracting appropriate unique normal forms. There is no rewriting theory involved, and the proof is completely constructive, allowing program extraction from the proof.

1 Introduction

In this paper we solve the decision problem for simply typed lambda calculus with categorical coproduct (strong disjoint sum) types. While this calculus is both natural and simple, the decision problem is a long-standing thorny issue in the subject. Our solution is based on normalization by evaluation (NBE) (also called "reduction-free normalisation") introduced by Martin-Löf [ML75] for weak typed lambda calculus, and by Berger and Schwichtenberg [BS91] for typed lambda calculus with $\beta\eta$-conversion. The technique has been further refined by the authors and coworkers using category-theoretic methods [CD97, AHS95, CDS97]. It has also been extended to other systems, such as System F [AHS96]. As shown by Berger, Eberl, Schwichtenberg, and Danvy [BES98, Da96], NBE techniques yield fast normalization algorithms, with applications in interactive proof systems [BBSSZ98] and type-directed partial evaluation [Da96, Da98, Fil01].

Here we show how to considerably extend the NBE techniques to take into account type systems with strong sums. The NBE method involves constructing a model $\mathcal{M}$ and effectively "inverting" the evaluation of lambda terms in $\mathcal{M}$ and thereby extracting certain unique syntactic normal forms, from which a decision procedure easily follows (we outline the proof below). The proof uses no rewriting theory.

Typed lambda calculi with (strong) sum types arise very naturally:

- In programming language theory, coproducts model variant and enumerative types. The added categorical equation for coproducts corresponds to a kind of uniqueness for *pattern matching* or `Case` construction [AC98, Mit96, GLT89].
- In proof theory, under the Curry-Howard Isomorphism, terms correspond to natural deduction proofs in intuitionistic propositional $\{\wedge, \vee, \Rightarrow, \top\}$ logic. One then considers terms (proofs) modulo certain equations, which guarantee, for example, that the formula $A \vee B$ acts as a coproduct type (with copairing), as well as including the theory of commutative conversions (cf [GLT89], pp 80-81). In category theoretic terminology, such lambda theories correspond exactly to *almost bicartesian closed categories*, that is, cartesian closed categories with nonempty finite coproducts (generated by a set of atomic types) [LS86].
- As proved by Dougherty and Subrahmanyam [DS95], a Friedman completeness theorem in **Set** holds for cartesian closed categories with binary coproducts. Therefore, the equality we decide is the natural extensional equality on proofs in intuitionistic propositional logic and on terms of the typed lambda calculus with sums.

Much of traditional lambda calculus theory carries through unscathed when we add products (and even weak categorical data types) to the simply typed case. Unfortunately, the addition of coproducts is considerably more subtle. The difficulties with adding coproducts are detailed in [Do93, DS95]: for example, the analog of Statman's 1-Section theorem fails in the presence of coproducts, confluence (of various standard rewriting

*School of Computer Science and IT, University of Nottingham, Nottingham, UK, *e-mail:* `txa@cs.nott.ac.uk`

†Department of Computing Science, Chalmers University of Technology, S-412 96 Göteborg, Sweden, *e-mail:* `peterd@cs.chalmers.se`

‡Division of Informatics (LFCS), University of Edinburgh, Edinburgh EH9 3JZ, UK, *e-mail:* `mxh@dcs.ed.ac.uk`

§Department of Mathematics, University of Ottawa, Ottawa, Ontario, K1N 6N5, Canada. *e-mail:* `phil@mathstat.uottawa.ca`.
The author's research is supported by a grant from NSERC.

presentations) fails, and the proof of Friedman's completeness theorem for the case of coproducts uses difficult and involved syntactical arguments [DS95].

A decision procedure for cartesian closed categories with binary coproducts has been presented in Ghani's thesis [Gh95a] (see [Gh95b] for a summary) although the proof involves intricate rewriting techniques whose details are daunting. Our method described here is quite different and we believe conceptually simpler.

An algorithm for type-directed partial evaluation for a call-by-value typed lambda calculus with sums has been given by Danvy [Da96, Da98] and Filinski [Fil01]. This algorithm uses continuations and is therefore also quite different from ours. In particular, it does not decide equality in cartesian closed categories with binary coproducts.

Like Ghani and Dougherty and Subrahmamyam, we only consider the case of finite *non-empty* coproducts, that is, an initial object (empty type) is not part of the structure. We conjecture that the present approach can be extended to full bicartesian closed categories including initial objects. However, this complicates the structure of our normal forms, and we have not yet completely checked that all properties hold for the extended language.

Outline of Proof

Let $\mathcal{E}$ be a lambda theory. Our aim is to decide if

$$\Gamma \vdash_{\mathcal{E}} e_1 = e_2 : A,$$

that is, if two possibly open terms e_1 and e_2 of type A are equal wrt $\mathcal{E}$, where Γ is a type environment . We associate with each term e a *normal form* $\mathsf{nf}(e)$. In this paper, these normal forms are not themselves terms, but there is a function d mapping normal forms to terms in such a way that the following two properties hold (cf. [CD97, CDS97]):

NF1 $\Gamma \vdash_{\mathcal{E}} \mathsf{d}(\mathsf{nf}(e)) = e$

NF2 $\Gamma \vdash_{\mathcal{E}} e_1 = e_2$ implies $\mathsf{nf}(e_1) = \mathsf{nf}(e_2)$.

This implies that $\Gamma \vdash_{\mathcal{E}} e_1 = e_2$ if and only if $\mathsf{nf}(e_1) = \mathsf{nf}(e_2)$, so that comparing normal forms will yield a decision procedure for $\mathcal{E}$.

When $\mathcal{E}$ = the typed lambda calculus with $\beta\eta$-conversion, the authors and coworkers showed in [AHS95, CDS97] how to obtain a function nf by inverting the presheaf interpretation of $\mathcal{E}$. One defines two natural transformations $\mathsf{q}^A : [\![A]\!] \to \mathrm{NF}(A)$ and $\mathsf{u}^A : \mathrm{NE}(A) \to [\![A]\!]$, where $\mathrm{NF}(A)$ is the presheaf of normal forms and $\mathrm{NE}(A)$ is the presheaf of neutral terms of type A from $\mathcal{E}$. Given a typing judgement $\Gamma \vdash_{\mathcal{E}} e : A$, where $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, we define

$$\mathsf{nf}(e) = \mathsf{q}([\![e]\!](\mathsf{u}(1_\Gamma)))$$

where 1_Γ is the sequence $(x_1, \ldots, x_n)$ and we omit type superscripts. Since $[\![-]\!]$ is an interpretation, we have immediately that $\Gamma \vdash e_1 = e_2$ implies $[\![e_1]\!] = [\![e_2]\!]$, and hence NF2 follows and NF1 is proved by induction on e, using for example logical relations.

How do we obtain a function nf when we add strong sums to $\mathcal{E}$? The problem is that although the category of presheaves has coproducts, a difficulty arises when we try to invert the interpretation of coproducts. The maps q and u are defined by induction on types, so in particular we need to define $\mathsf{u}^{A_0+A_1}$ in terms of u^{A_0} and u^{A_1}. But coproducts in presheaves are calculated pointwise; so, for example, how do we define $\mathsf{u}^{A_0+A_1}(s) \in [\![A_0]\!]_\Gamma + [\![A_1]\!]_\Gamma$ for a neutral term $\Gamma \vdash s : A_0 + A_1$? Since variables are neutral terms, we must in particular define $\mathsf{u}^{A_0+A_1}(x)$, but there is no sensible way to decide whether this should be in the first or the second disjunct.

As we shall show, the solution of this problem is to introduce an appropriate Grothendieck topology and consider the sheaves for that topology. This will give us a way to "amalgamate" the contributions of u^{A_0} and u^{A_1} in the definition of $\mathsf{u}^{A_0+A_1}$.

Plan of the paper

In Section 2 we formally define the typed lambda calculus with strong sums and show how it yields a free cartesian closed category with binary coproducts. In Section 3 we introduce our normal forms, and the auxiliary notions of pure normal forms and neutral terms. The main idea is to introduce a parallel case statement, and impose variable conditions and a condition of redundancy-freeness to obtain uniqueness of normal forms. In Section 4 we introduce the category of constrained environments, where objects are environments (type assigments) equipped with equational constraints. This will serve as the underlying category of our Grothendieck topology which is defined in Section 5. There we also introduce the category of sheaves for this topology and its bicartesian closed structure. This yields a canonical interpretation of the syntax in the category of sheaves and in Section 6 we show how to invert this interpretation and obtain normal forms.

2 Syntax

We follow the treatment of sums in natural deduction, as in [GLT89, pp 80-81]. For ease of presentation, we restrict ourselves to one base type.

Types are given by the grammar

$$A ::= o \mid A \Rightarrow A \mid A \times A \mid \top \mid A + A$$

Terms are given by

$$e \quad ::= \quad x \mid \lambda x.e \mid e\,e \mid \langle e, e\rangle \mid \pi_0(e) \mid \pi_1(e) \mid \langle\rangle \mid \iota_0(e) \mid \iota_1(e) \mid \delta\ (x.e)\ (x.e)\ e$$

The `Case` term $\delta\ (x_0.e_0)\ (x_1.e_1)\ e_2$ simultaneously binds x_0 in e_0 and x_1 in e_1.

A *type environment* Γ is a finite function from variables to types. The typing judgement $\Gamma \vdash e : A$ meaning *e has type A in type environment* Γ is defined in the obvious way. For example, the rule for `Case` is:

$$\frac{(\Gamma, x_i : A_i \vdash e_i : C)_{i\in\{0,1\}} \qquad \Gamma \vdash e : A_0 + A_1}{\Gamma \vdash \delta\ (x_0.e_0)\ (x_1.e_1)\ e : C}$$

Definition 2.1 *Equality between terms in environment* Γ, denoted $\Gamma \vdash - = - : A$, is the least (typed) congruence generated by the following rules (omitting types to improve readability):

(β)	$(\lambda x.e_0)e_1 = e_0[e_1/x]$
(η)	$e = \lambda x.e\,x$, if $x \notin \mathrm{FV}(e)$
Proj$_i$	$\pi_i(\langle e_0, e_1\rangle) = e_i$
SP	$e = \langle \pi_0(e), \pi_1(e)\rangle$
Unit	$e = \langle\rangle$
In$_i$	$\delta\ (x_0.e_0)\ (x_1.e_1)\ \iota_i(e_2) = e_i[e_2/x_i]$
Coprod	$\delta\ (x_0.\iota_0(x_0))\ (x_1.\iota_1(x_1))\ e = e$
Distrib	$e\,(\delta\ (x_0.e_0)\ (x_1.e_1)\ e_2) = \delta\ (x_0.e\,e_0)\ (x_1.e\,e_1)\ e_2$ if $x_0, x_1 \notin \mathrm{FV}(e)$

We will refer to this equational theory as BiCCC. The key categorical axiom (Coprod) is dual to (SP) and guarantees uniqueness of the co-pairing arrow out of a coproduct. BiCCC entails all the usual commutative conversions for sums, [GLT89], pp. 80-81.

It can be shown (cf. [LS86, CDS97]) that the free almost bicartesian closed category $\mathcal{B}_0$ over one base object o can be obtained as the category whose objects are type environments and where a morphism from $\Gamma = x_1 : A_1, \ldots, x_m : A_m$ to $\Delta = y_1 : B_1, \ldots, y_n : B_n$ is a sequence of terms $(e_1, \ldots, e_n)$, modulo BiCCC equality, where $\Gamma \vdash e_i : B_i$. *Freeness* means that for each BiCCC $\mathcal{B}$ and object $[\![o]\!] \in \mathcal{B}$ we have a unique structure- and equation-preserving interpretation functor $[\![-]\!] : \mathcal{B}_0 \to \mathcal{B}$.

3 Normal Forms

Normal forms are defined simultaneously with *pure normal forms* and *neutral terms*. Normal (and pure normal) forms are not genuine terms, but defined inductively by the clauses below. If Γ is a type environment we write $\Gamma \vdash_{\mathrm{NF}} t : A$, resp. $\Gamma \vdash_{\mathrm{PNF}} t : A$, resp. $\Gamma \vdash_{\mathrm{NE}} t : A$ to mean that expression t is a normal form, resp. pure normal form, resp. neutral term of type A. We write $\mathrm{FV}(t)$ for the set of free variables occurring in t. We write $\mathrm{Guards}(t)$ for the set of *guards* of a normal form t; this will be defined below as part of the rule for forming normal forms.

$$\frac{x \in \mathrm{dom}(\Gamma)}{\Gamma \vdash_{\mathrm{NE}} x : \Gamma(x)} \qquad \frac{\Gamma \vdash_{\mathrm{NE}} s : o}{\Gamma \vdash_{\mathrm{PNF}} s : o}$$

$$\Gamma \vdash_{\mathrm{PNF}} \langle\rangle : \top$$

$$\frac{\Gamma \vdash_{\mathrm{PNF}} t_0 : A_0 \qquad \Gamma \vdash_{\mathrm{PNF}} t_1 : A_1}{\Gamma \vdash_{\mathrm{PNF}} \langle t_0, t_1\rangle : A_0 \times A_1}$$

$$\frac{\Gamma \vdash_{\mathrm{NE}} t : A_0 \times A_1}{\Gamma \vdash_{\mathrm{NE}} \pi_i(t) : A_i} \quad i \in \{0,1\}$$

$$\frac{\Gamma \vdash_{\mathrm{PNF}} t : A_i}{\Gamma \vdash_{\mathrm{PNF}} \iota_i(t) : A_0 + A_1} \quad i \in \{0,1\}$$

$$\frac{\Gamma \vdash_{\mathrm{NE}} s : A \Rightarrow B \qquad \Gamma \vdash_{\mathrm{PNF}} t : A}{\Gamma \vdash_{\mathrm{NE}} st : B}$$

$$\frac{\Gamma, x{:}A \vdash_{\mathrm{NF}} t : B}{\Gamma \vdash_{\mathrm{PNF}} \lambda x.t : A \Rightarrow B}$$

where in the last rule we have the variable condition that $x \in \mathrm{FV}(s)$ for each $s \in \mathrm{Guards}(t)$.

We have two rules for forming normal forms:

(a) $$\frac{\Gamma \vdash_{\mathrm{PNF}} t : A}{\Gamma \vdash_{\mathrm{NF}} t : A} \quad \text{and} \quad \mathrm{Guards}(t) = \emptyset$$

(b) Let $M = \{s_1, \ldots, s_n\}$ be a nonempty finite set of neutral terms (so we assume the s_i are pairwise distinct). For each $f : M \to \{0,1\}$ we use the abbreviation $\Gamma_f = \Gamma, x_1{:}A^1_{f(s_1)}, \ldots, x_n{:}A^n_{f(s_n)}$. Define

$$\frac{\begin{array}{c}(\Gamma \vdash_{\mathrm{NE}} s_i : A^i_0 + A^i_1)_{i\in\{1,\ldots,n\}} \\ (\Gamma_f \vdash_{\mathrm{NF}} t_f : C)_{f:M\to\{0,1\}}\end{array}}{\Gamma \vdash_{\mathrm{NF}} \mathcal{C}(M, (x_1 \cdots x_n.t_f)_f) : C}$$

and $\qquad \mathrm{Guards}(\mathcal{C}(M, (x_1 \cdots x_n.t_f)_f)) = M$

where $(t_f)_{f:M\to\{0,1\}}$ is a family of normal forms satisfying the following two side conditions:

Variable-condition: for each $s \in \mathrm{Guards}(t_f)$ we have $\{x_1, \ldots, x_n\} \cap \mathrm{FV}(s) \neq \emptyset$.

Redundancy-freeness: The family $(t_f)_f$ is not redundant at any $s_i \in M$, where $(t_f)_f$ is redundant at s_i whenever for all $g : M \setminus \{s_i\} \to \{0,1\}$, $t_{g[s_i\mapsto 0]}$ and $t_{g[s_i\mapsto 1]}$ are equal and neither contains the variable x_i.

The variables $x_1, \ldots, x_n$ become bound in the $\mathcal{C}$-construct. For brevity we shall often use the alternative notation $\mathcal{C}(M, (t'_f)_f)$, where the t'_f range over abstractions $x_1, \ldots, x_n.t_f$.

The idea is that $\mathcal{C}$ performs a simultaneous case split over all the "guards". For example, $t_{f[s\mapsto 0]}$ corresponds to a branch to be taken when s is of the form $\iota_0(x)$.

Example 3.1 The following examples show how the side-conditions ensure uniqueness of normal forms as computed by nf in Section 1. For simplicity let the variables z (possibly with indices) in the examples below have type o, so that they are normal terms.

1. The normal form of $\lambda w.\delta\ (x_1.z_0)\ (x_1.z_1)\ y$ will be $\mathcal{C}(\{y\}, (x_1.t_f)_f)$ where $t_{[y\mapsto i]} = \lambda w.z_i$. Note that the expression $\lambda w.\mathcal{C}(\{y\}, (x_1.t_f)_f)$, where $t_{[y\mapsto i]} = z_i$, violates the side condition for (pure) normal forms of λ-form.

2. The normal form of the term

$$\begin{aligned}&\delta\ (x_1.\delta\ (x_2.z_{00})\ (x_2.z_{01})\ y_2)\\&\quad(x_1.\delta\ (x_2.z_{10})\ (x_2.z_{11})\ y_2)\\&\quad y_1\end{aligned}$$

   will be

$$\mathcal{C}(\{y_1, y_2\}, (x_1 x_2.t_f)_f)$$

   where $t_{[y_1\mapsto i, y_2\mapsto j]} = z_{ij}$. Note that the expression $\mathcal{C}(\{y_1\}, (x_1.\mathcal{C}(\{y_2\}, (x_2.t_{f_1\cup f_2})_{f_2})_{f_1}))$ is not a normal form since it violates the variable-condition: x_1 is not free in the guard y_2 of the normal form $\mathcal{C}(\{y_2\}, (x_2.t_{f_1\cup f_2})_{f_2})$.

3. The normal form of $\delta\ (x.z)\ (x.z)\ y$ will be z. Note that $\mathcal{C}(\{y\}, (x.z)_f)$ is not a normal form as $(x.z)_f$ is redundant at y.

4. Note however, that the normal form of $\delta\ (z.z)\ (z.z)\ y$ will be $\mathcal{C}(\{y\}, (z.z)_f)$ which is not redundant at y because of the variable condition in the definition of redundancy.

Definition 3.2 The function d mapping $\Gamma \vdash_X t : A$ with $X \in \{\mathrm{NF}, \mathrm{PNF}, \mathrm{NE}\}$ to terms $\Gamma \vdash \mathsf{d}(t) : A$ is defined in the following way:

- d commutes with all the term formers except $\mathcal{C}$ (in particular, preserves variables).
- $\mathsf{d}(\mathcal{C}(M \cup \{s\}, (t_f)_f)) = \delta\ (x_0.e_0)\ (x_1.e_1)\ \mathsf{d}(s)$, where $e_i = \mathsf{d}(\mathcal{C}(M, (t_{g[s\mapsto i]})_g))$.

It is easy to see that up to BiCCC equality this does not depend on the choice of the witnessing term e_Γ and on the order of the guards.

4 Neutral constrained environments

Like Dougherty and Subrahmanyam [DS95] and Fiore and Simpson [FS99] we need to supply our type environments with constraints. These will be the objects of a category of constrained environments $\mathcal{N}$, where the morphisms will be injective renamings. The constraints are of the form $s = \iota_i(x_i)$ and express which branch a certain guard s takes. This is the idea behind our Grothendieck topology on $\mathcal{N}$: a "covering" expresses case-splitting. This use of Grothendieck topologies is closely related to [FS99] where they were used for proving a definability result for a language with coproducts.

Definition 4.1 A *neutral constrained environment*, environment for short, is a pair $\Gamma \mid \Xi$ where Γ is a type environment and Ξ is a set of constraints of the form $s = \iota_0(x_0)$ or $s = \iota_1(x_1)$ where $\Gamma \vdash_{\mathrm{NE}} s : A_0 + A_1$ and $x_0 : A_0$ (resp. $x_1 : A_1$) is contained in Γ and moreover,

- no two distinct constraints involve the same neutral term, for example, Ξ cannot contain $s{=}\iota_0(x_0)$ and $s{=}\iota_1(x_1)$
- no two distinct constraints refer to the same variable, for example, Ξ cannot contain $s = \iota_0(x_0)$ and $s' = \iota_0(x_0)$ unless s and s' are identical.

A *morphism* from environment $\Delta \mid \Psi$ to environment $\Gamma \mid \Xi$ is given by an *injective* function $\sigma : \mathrm{dom}(\Gamma) \to \mathrm{dom}(\Delta)$ satisfying $\Delta(\sigma(x)) = \Gamma(x)$ and $\sigma(s){=}\iota_i(\sigma(x))$ is in Ψ for each constraint $s{=}\iota_i(x)$ in Ξ. In this way the environments form a category $\mathcal{N}$ in which composition is composition of functions.

If Δ extends Γ and Ψ extends Ξ then the inclusion $\sigma : \mathrm{dom}(\Gamma) \hookrightarrow \mathrm{dom}(\Delta)$ defines a morphism from $\Delta \mid \Psi$ to $\Gamma \mid \Xi$ which we call a *projection*.

We are interested in studying equality of terms relative to a neutral constrained environment. The following definition is due to [DS95].

Definition 4.2 Let $\Gamma \mid \Xi$ be an environment and $\vec{d}$ be a list of dummy terms of the same length as Ξ and of appropriate (to be explained) type. A (variable-binding) type environment $C^{\Gamma \mid \Xi}_{\vec{d}}[\,]$ is defined as follows.

$$C^{\Gamma \mid \emptyset}[\,] = [\,]$$

$$C^{\Gamma, x_0:A_0 \mid \Xi, s=\iota_0(x_0)}_{\vec{d}, d_1}[\,] = \delta\ (x_0.C^{\Gamma \mid \Xi}_{\vec{d}}[\,])\ (x_1.d_1 x_1)\ \mathsf{d}(s)$$

$$C^{\Gamma, x_1:A_1 \mid \Xi, s=\iota_1(x_1)}_{\vec{d}, d_0}[\,] = \delta\ (x_0.d_0 x_0)\ (x_1.C^{\Gamma \mid \Xi}_{\vec{d}}[\,])\ \mathsf{d}(s)$$

Note that $C^{\Gamma \mid \Xi}_{\vec{d}}[e]$ binds all variables mentioned in Ξ.

Given two terms $\Gamma \vdash e_1 : C$ and $\Gamma \vdash e_2 : C$ we write $\Gamma \,|\, \Xi \vdash e_1 = e_2 : C$ to mean that

$$\Gamma' \vdash C^{\Gamma \,|\, \Xi}_{\vec{d}}[e_1] = C^{\Gamma \,|\, \Xi}_{\vec{d}}[e_2] : C$$

in the theory BiCCC for all appropriate Γ' and $\vec{d}$. Here $\vec{d}$ must be chosen such that the terms $C^{\Gamma \,|\, \Xi}_{\vec{d}}[e_i]$ are type correct and Γ' is obtained from Γ by removing the variables mentioned in Ξ and possibly adding any extra free variables occurring in the dummy terms $\vec{d}$.

Remark 4.3 Note that ordinary type environments have no constraints but it follows immediately from the above definition that $\Gamma|\emptyset \vdash e_1 = e_2$ implies $\Gamma \vdash e_1 = e_2$.

5 Sheaves over environments

We consider the functor category $\widehat{\mathcal{N}} \stackrel{\text{def}}{=} \mathit{Sets}^{\mathcal{N}^{\text{op}}}$ of presheaves and natural transformations between them. We recall the following definitions of the structure of $\widehat{\mathcal{N}}$. A *presheaf* is given by a family of sets $F_{\Gamma \,|\, \Xi}$ indexed by environments and for each morphism $\sigma : \Delta \,|\, \Psi \to \Gamma \,|\, \Xi$ a function $F_\sigma : F_{\Gamma \,|\, \Xi} \to F_{\Delta \,|\, \Psi}$ such that $F_1 = 1$ and $F_{\sigma \circ \tau} = F_\tau \circ F_\sigma$. If $a \in F_{\Gamma \,|\, \Xi}$ we may write $a\lceil_{\Delta \,|\, \Psi}$ for $F_\sigma(a)$ in case σ is clear from the context. This notation will in particular be used when σ is a projection.

A *natural transformation* from presheaf F to presheaf G is given by a family $g_{\Gamma \,|\, \Xi}$ of maps $g_{\Gamma \,|\, \Xi} : F_{\Gamma \,|\, \Xi} \to G_{\Gamma \,|\, \Xi}$ such that $G_\sigma \circ g_{\Gamma \,|\, \Xi} = g_{\Delta \,|\, \Psi} \circ F_\sigma$ (*naturality*). If $a \in F_{\Gamma \,|\, \Xi}$ we may write $g(a)$ for $g_{\Gamma \,|\, \Xi}(a)$. Naturality then reads $g(a)\lceil_{\Delta \,|\, \Psi} = g(a\lceil_{\Gamma \,|\, \Xi})$.

As any category of presheaves, the category $\widehat{\mathcal{N}}$ is bicartesian closed, that is, supports the interpretation of the type formers $\top, \times, \Rightarrow, +$, (and $\perp$). If we denote the interpreting presheaves with the same symbols thus writing e.g. $F \Rightarrow G$ for the function space of presheaves, we have the following explicit constructions of the type formers in $\mathit{Sets}^{\mathcal{N}^{\text{op}}}$:

$$\begin{array}{lcl} \top_{\Gamma \,|\, \Xi} & = & \{\langle\rangle\} \\ (F \times G)_{\Gamma \,|\, \Xi} & = & F_{\Gamma \,|\, \Xi} \times G_{\Gamma \,|\, \Xi} \\ (F + G)_{\Gamma \,|\, \Xi} & = & F_{\Gamma \,|\, \Xi} + G_{\Gamma \,|\, \Xi} \\ (F \Rightarrow G)_{\Gamma \,|\, \Xi} & = & \widehat{\mathcal{N}}(\mathcal{N}(-, \Gamma \,|\, \Xi) \times F\ G) \end{array}$$

However, as we mentioned in the introduction, we are not able to obtain normal forms by inverting this presheaf interpretation. Instead we shall consider the interpretation of terms in the category of sheaves over a certain topology, and show that this can be inverted.

Recall that the basis of a *Grothendieck topology* is a collection of *basic coverings*, satisfying the axioms of identity, transitivity, and stability [MM92, p.111]. A covering of an object $\Gamma \,|\, \Xi$ in $\mathcal{N}$ is here a family of arrows with codomain $\Gamma \,|\, \Xi$. Since the category $\mathcal{N}$ does not have pullbacks in general, we use a modified axiom of stability [MM92, p.156]. Moreover, like [FS99] we only require that the identity is a singleton covering, not that all isomorphisms are coverings.

Definition 5.1 The basis K for a Grothendieck topology on $\mathcal{N}$ is inductively generated by the following clauses:

- The identity covering containing only the arrow $1_{\Gamma \,|\, \Xi}$ is a basic covering of $\Gamma \,|\, \Xi$.
- If $\Gamma \vdash_{\text{NE}} s : A_0 + A_1$ and s is not mentioned in Ξ, and if the family of projections from $(\Gamma_i \,|\, \Xi_i)_i$ forms a basic covering of $\Gamma, x_0 : A_0|\Xi, s = \iota_0(x_0)$ and the family of projections from $(\Gamma_j \,|\, \Xi_j)_j$ forms a basic covering of $\Gamma, x_1 : A_1|\Xi, s = \iota_1(x_1)$, then the disjoint union of the projections from $(\Gamma_i \,|\, \Xi_i)_i$ and $(\Gamma_j \,|\, \Xi_j)_j$ forms a basic covering of $\Gamma \,|\, \Xi$.

The general concept of sheaves for Grothendieck topologies need not be presented, since it here specialises to the following rather digestible definition:

Proposition 5.2 *A presheaf F is a* sheaf *for K iff whenever $\Gamma \,|\, \Xi$ is covered by $\Gamma, x_0{:}A_0 \,|\, \Xi, s{=}\iota_0(x_0)$ and $\Gamma, x_1{:}A_1 \,|\, \Xi, s{=}\iota_1(x_1)$, that is, $\Gamma \vdash_{\text{NE}} s : A_0 + A_1$ and*

$$\begin{array}{lcl} f_0 & \in & F_{\Gamma, x_0:A_0 \,|\, \Xi, s=\iota_0(x_0)} \\ f_1 & \in & F_{\Gamma, x_1:A_1 \,|\, \Xi, s=\iota_1(x_1)} \end{array}$$

then there exists a unique *$f \in F_{\Gamma \,|\, \Xi}$ (called* pasting*) such that*

$$\begin{array}{lcl} f\lceil_{\Gamma, x_0:A_0 \,|\, \Xi, s=\iota_0(x_0)} & = & f_0 \\ f\lceil_{\Gamma, x_1:A_1 \,|\, \Xi, s=\iota_1(x_1)} & = & f_1 \end{array}$$

The following result follows from general properties of Grothendieck topologies and will therefore not be proved, see [MM92] for an exposition.

Proposition 5.3

1. *The terminal object in $\widehat{\mathcal{N}}$ is a sheaf,*
2. *if F, G are sheaves so is $F \times G$ (cartesian product),*
3. *if G is a sheaf and F is a presheaf then $F \Rightarrow G$ is a sheaf (function space)*
4. *for each presheaf F there exists a sheaf aF (the* associated sheaf *or* sheafification*) and a natural transformation $\eta : F \to aF$ such that whenever G is a sheaf and $f : F \to G$ then there exists a unique $f^\sharp : aF \to G$ with $f^\sharp \circ \eta = f$. In other words, the sheaves form a reflective subcategory of $\mathcal{N}$,*
5. *The sheafification functor a preserves binary products.*

6. *if F, G are sheaves the coproduct $F + G$ is in general not a sheaf, but $a(F + G)$ is the coproduct of F and G in the subcategory of sheaves.*

7. *if $u, v : F \to G$ and F, G are sheaves then the equaliser of u and v is a sheaf.*

We write $\Gamma \,|\, \Xi \vdash_{\mathrm{NF}} t : A$ to mean that $\Gamma \vdash_{\mathrm{NF}} t : A$ and, moreover, none of the neutral terms mentioned in Ξ is contained in $\mathrm{Guards}(t)$. Intuitively, this is because no case split is ever needed for a guard whose value is already known through the environment. Note that there is no need to define $\Gamma \,|\, \Xi \vdash_{\mathrm{NE}} t : A$ and $\Gamma \,|\, \Xi \vdash_{\mathrm{PNF}} t : A$, since all guards inside neutral and pure normal terms include variables bound by λ's. Hence the constraints in Ξ cannot affect t.

For a type A we define the presheaves $\mathrm{NF}(A)$, $\mathrm{PNF}(A), \mathrm{NE}(A), \mathrm{Term}(A)$ as follows:

$$\begin{array}{rcl} \mathrm{NF}(A)_{\Gamma \,|\, \Xi} & = & \{t \mid \Gamma \,|\, \Xi \vdash_{\mathrm{NF}} t : A\} \\ \mathrm{PNF}(A)_{\Gamma \,|\, \Xi} & = & \{t \mid \Gamma \vdash_{\mathrm{PNF}} t : A\} \\ \mathrm{NE}(A)_{\Gamma \,|\, \Xi} & = & \{t \mid \Gamma \vdash_{\mathrm{NE}} s : A\} \\ \mathrm{Term}(A)_{\Gamma \,|\, \Xi} & = & \{t \mid \Gamma \,|\, \Xi \vdash t : A\}/\!\sim_{\vdash} \end{array}$$

where $t \sim_{\vdash} t'$ stands for $\Gamma \,|\, \Xi \vdash t = t' : A$.

If $\sigma : \Delta \,|\, \Psi \to \Gamma \,|\, \Xi$ and $\Gamma \,|\, \Xi \vdash_{\mathrm{NE}} t : A$ then $\mathrm{NE}(A)_\sigma(t) \in \mathrm{NE}(A)_{\Delta \,|\, \Psi}$ is defined by replacing each free variable x in t by $\sigma(x)$. The morphism parts Term_σ and PNF_σ are defined analogously.

If $t \in \mathrm{NF}_{\Gamma \,|\, \Xi}(A)$ then $\mathrm{NF}_\sigma(t)$ is defined by first replacing each free variable x in t by $\sigma(x)$ and then plugging in all the constraints mentioned in Ψ by repeatedly performing the following atomic restriction operation (an analogous operation appears in Ghani's thesis [Gh95a] under the name "first and second residue").

Definition 5.4 Let $t \in \mathrm{NF}(C)_{\Gamma \,|\, \Xi}$ and $\Gamma \vdash_{\mathrm{NE}} s : A_0 + A_1$. Then we define the restriction $t[s{:=}\iota_i(x_i)]$ of t to $\Gamma, x_i : A_i | \Xi, s{=}\iota_i(x_i)$ (along the projections) as follows.

$$\begin{array}{rcl} t[s{:=}\iota_i(x)] & = & t, \text{if } s \notin \mathrm{Guards}(t) \\ \mathcal{C}(M \cup \{s\}, (t_f)_f)[s{:=}\iota_i(x_i)] & = & \mathcal{C}^{\mathrm{nf}}(M, (t_{g[s\mapsto i]})_g) \end{array}$$

where $\mathcal{C}^{\mathrm{nf}}$ computes a normal form to be defined below. Note that we cannot simply replace $\mathcal{C}^{\mathrm{nf}}$ by $\mathcal{C}$ because the set of guards can become empty upon plugging in a constraint, new redundancies may be created, and the variable conditions may be violated. We define $\mathcal{C}^{\mathrm{nf}}(\emptyset, \{t\})$ to be t and $\mathcal{C}(M \cup \{s\}, (t_f)_f)$ to be $\delta^{\mathrm{nf}}\ (x_0.\mathcal{C}^{\mathrm{nf}}(M, (t_{f[s\mapsto 0]})_f))\ (x_1.\mathcal{C}^{\mathrm{nf}}(M, (t_{f[s\mapsto 1]})))\ s$.

To compute $\delta^{\mathrm{nf}}\ (x_0.t_0)\ (x_1.t_1)\ s$ we first check whether t_i depend on x_i and are different (see the definition of redundancy). If not, we return $t_0 (= t_1)$, or otherwise, we return $\mathcal{C}(\{s\} \cup M_0 \cup M_1, t_g)$, where

$$M_i = \{s_i \in \mathrm{Guards}(t_i) | x_i \notin \mathrm{FV}(s_i)\}$$

for $i = 0, 1$, and the family t_g is adjusted accordingly.

Proposition 5.5 d *defines natural transformations* $\mathrm{NF}(A) \to \mathrm{Term}(A)$, $\mathrm{PNF}(A) \to \mathrm{Term}(A)$, $\mathrm{NE}(A) \to \mathrm{Term}(A)$.

If $f : \mathcal{B}(\Delta, \Gamma)$ is a morphism in the free BiCCC $\mathcal{B}$, that is, a sequence of terms in type environment Δ, then $[t] \mapsto [ft]$ defines a natural transformation $\mathrm{Term}(f) : \mathrm{Term}(\Delta) \to \mathrm{Term}(\Gamma)$. This makes $\mathrm{Term}(-)$ a functor from $\mathcal{B}$ to $\widehat{\mathcal{N}}$ preserving $\top$ and cartesian products.

Proposition 5.6 *The presheaf* $\mathrm{Term}(A)$ *is a sheaf.*

Proposition 5.7 *The presheaf* $\mathrm{NF}(A)$ *is a sheaf and is isomorphic to the sheafification* $a(\mathrm{PNF}(A))$ *of* $\mathrm{PNF}(A)$ *with the embedding* $\eta : \mathrm{PNF}(A) \to \mathrm{NF}(A)$ *given by* $\eta_{\Gamma \,|\, \Xi}(t) = t$.

If $\Gamma \vdash s : A_0 + A_1$, then the pasting of two normal forms $t_i \in \mathrm{NF}(A)_{\Gamma, x_i : A_i | \Xi, s = \iota_i(x_i)}$ is the normal form $\delta^{\mathrm{nf}}(x_0.t_0)(x_1.t_1)s \in \mathrm{NF}(A)_{\Gamma \,|\, \Xi}$.

Let us write $\mathit{Sh}(\mathcal{N})$ for the full subcategory of $\widehat{\mathcal{N}}$ consisting of the sheaves. We know from Prop. 5.3 that $\mathit{Sh}(\mathcal{N})$ is a BiCCC. Since the category $\mathcal{B}_0$ of sequences of types and terms is a free BiCCC there is a unique interpretation functor $[\![-]\!] : \mathcal{B}_0 \to \mathit{Sh}(\mathcal{N})$, determined by

$$[\![o]\!] = \mathrm{NF}(o)$$

Concretely, this functor is given by defining a canonical BiCCC structure on $\mathit{Sh}(\mathcal{N})$.

6 Inverting the interpretation function

We will now define natural transformations

$$\begin{array}{l} \mathsf{q}^A : [\![A]\!] \to \mathrm{NF}(A) \\ \mathsf{u}^A : \mathrm{NE}(A) \to [\![A]\!] \end{array}$$

in such a way that for a term $\Gamma \vdash e : A$,

$$\mathsf{nf}(e) \stackrel{\mathrm{def}}{=} \mathsf{q}^A_\Gamma([\![e]\!](\mathsf{u}^\Gamma_\Gamma(1_\Gamma)))$$

will satisfy NF1:

- $\mathsf{q}^o : \mathrm{NF}(o) \to \mathrm{NF}(o)$ is the identity function.

  $\mathsf{u}^o : \mathrm{NE}(o) \to \mathrm{NF}(o)$ is the injection from neutral terms to normal terms given by the obvious term-formation rules.

- $\mathsf{q}^\top : \top \to \mathrm{NF}(\top)$ is the constant function returning the normal form $\langle\rangle$.

  $\mathsf{u}^\top : \mathrm{NE}(\top) \to \top$ is the constant function returning the element $\langle\rangle \in \top$. (As before we use the same signs for corresponding syntactic and semantic notions.)

- $\mathsf{q}^{A_0\times A_1} = \mathrm{pair}^{\mathrm{nf}}\circ(\mathsf{q}^{A_0}\times\mathsf{q}^{A_1})$ where $\mathrm{pair}^{\mathrm{nf}} : \mathrm{NF}(A_0)\times\mathrm{NF}(A_1)\to\mathrm{NF}(A_0\times A_1)$ is the unique map satisfying $\mathrm{pair}^{\mathrm{nf}}(t_1,t_2) = \langle t_1,t_2\rangle$ for *pure* normal forms t_1,t_2. This map exists by Proposition 5.7 and the fact that a preserves products.

$$\mathsf{u}^{A_0\times A_1}_{\Gamma\,|\,\Xi}(s) = \langle \mathsf{u}^{A_0}_{\Gamma\,|\,\Xi}(\pi_0(s)), \mathsf{u}^{A_1}_{\Gamma\,|\,\Xi}(\pi_1(s))\rangle$$

- Let $\theta \in [\![A \Rightarrow B]\!]_{\Gamma\,|\,\Xi} = \widehat{\mathcal{N}}(\mathcal{N}(-,\Gamma\,|\,\Xi)\times[\![A]\!], [\![B]\!])$. Then

$$\mathsf{q}^{A\Rightarrow B}_{\Gamma\,|\,\Xi}(\theta) = \lambda^{\mathrm{nf}}x.\mathsf{q}^{B}_{\Gamma,x:A|\Xi}(\theta(\sigma,\mathsf{u}^{A}_{\Gamma,x:A|\Xi}(x)))$$

where σ is the projection from $\Gamma, x : A|\Xi$ to $\Gamma\,|\,\Xi$. Here $\lambda^{\mathrm{nf}}x.\mathcal{C}(M,(x_1\dots x_n.t_f)_f)$ is obtained by dividing M into two sets, M_0 which contains the guards which do not depend on x, and M_1, which contains the guards which do. Then we return

$$\mathcal{C}(M_0,(x_1\dots x_{n_0}.\lambda x.\mathcal{C}^{\mathrm{nf}}(M_1,(x_1\dots x_{n_1}.t_{f_0\cup f_1})_{f_1}))_{f_0})$$

Compare also example 1 in 3.1.

Let $s \in \mathrm{NE}(A \Rightarrow B)_{\Gamma\,|\,\Xi}$. Then $\mathsf{u}^{A\Rightarrow B}_{\Gamma\,|\,\Xi}(s) \in [\![A\Rightarrow B]\!]_{\Gamma\,|\,\Xi}$ is defined by

$$(\mathsf{u}^{A\Rightarrow B}_{\Gamma\,|\,\Xi}(s))_{\Delta\,|\,\Psi}(\sigma,a) =$$
$$\mathsf{u}^{B}_{\Delta\,|\,\Psi}(\mathrm{NE}_\sigma(s)(\mathsf{q}^{A}_{\Delta\,|\,\Psi}(a))) \in [\![B]\!]_{\Delta\,|\,\Psi}$$

where $\sigma \in \mathcal{N}(\Delta\,|\,\Psi,\Gamma\,|\,\Xi)$ and $a \in [\![A]\!]_{\Delta\,|\,\Psi}$.

- $\mathsf{q}^{A_0+A_1}$ is the unique map (arising from the coproduct property of $[\![A_0+A_1]\!]$) satisfying

$$\mathsf{q}^{A_0+A_1}(\iota_0^{\mathrm{sh}}(a)) = \iota_0^{\mathrm{nf}}(\mathsf{q}^{A_0}(a))$$
$$\mathsf{q}^{A_0+A_1}(\iota_1^{\mathrm{sh}}(b)) = \iota_1^{\mathrm{nf}}(\mathsf{q}^{A_1}(b))$$

Here $\iota_0^{\mathrm{sh}}, \iota_1^{\mathrm{sh}}$ are the coproduct injections in $Sh(\mathcal{N})$ and $\iota_0^{\mathrm{nf}} : \mathrm{NF}(A_0)\to\mathrm{NF}(A_0+A_1)$ is the unique map satisfying $\iota_0^{\mathrm{nf}}(t) = \iota_0(t)$ for *pure* normal form $t : A_0$. Similarly for ι_1^{nf}.

To construct

$$\mathsf{u}^{A_0+A_1} \in \mathrm{NE}(A_0+A_1)\to[\![A_0+A_1]\!]$$

consider $s \in \mathrm{NE}(A_0+A_1)_{\Gamma\,|\,\Xi}$: either $s = \iota_0(x) \in \Xi$ in which case we put $f_{\Gamma\,|\,\Xi}(s) = \iota_0^{\mathrm{sh}}(\mathsf{u}^{A_0}_{\Gamma\,|\,\Xi}(x))$, or $s = \iota_1(y) \in \Xi$ and we put $f_{\Gamma\,|\,\Xi}(s) = \iota_1^{\mathrm{sh}}(\mathsf{u}^{A_1}_{\Gamma\,|\,\Xi}(y))$, or s is not mentioned in Ξ in which case we define $f_{\Gamma\,|\,\Xi}(s)$ as the unique pasting of

$$a_0 \stackrel{\mathrm{def}}{=} \iota_0^{\mathrm{sh}}(\mathsf{u}^{A_0}_{\Gamma,x:A_0\,|\,\Xi,s=\iota_0(x)}(x))$$
$$a_1 \stackrel{\mathrm{def}}{=} \iota_1^{\mathrm{sh}}(\mathsf{u}^{A_1}_{\Gamma,x:A_1\,|\,\Xi,s=\iota_1(x)}(x))$$

It follows by straightforward calculations that all these are indeed natural transformations.

Proposition 6.1 *In order to establish NF1, that is,* $e = \mathsf{d}(\mathsf{q}^A([\![e]\!](\mathsf{u}(1_\Gamma))))$ *for* $\Gamma \vdash e : A$ *we define a family of subsheaves* $R^A_{\Gamma|\Xi} \subseteq [\![A]\!]_{\Gamma|\Xi}\times\mathrm{Term}(A)_{\Gamma\,|\,\Xi}$*, such that*

(i) *For all* $a \in [\![A]\!]_{\Gamma\,|\,\Xi}$ *and* $\Gamma \vdash e : A$*:*

$$aR^A_{\Gamma\,|\,\Xi}e \Rightarrow \Gamma\,|\,\Xi \vdash \mathsf{d}(\mathsf{q}^A_{\Gamma\,|\,\Xi}(a)) = e$$

(ii) *For all* $s \in \mathrm{NE}(A)_{\Gamma\,|\,\Xi}$

$$\mathsf{u}^A_{\Gamma\,|\,\Xi}(s)R^A_{\Gamma\,|\,\Xi}\mathsf{d}(s)$$

We can extend R to type environments by letting $(a_1,\dots,a_n)\ R^\Gamma_{\Gamma\,|\,\Xi}\ (f_1,\dots,f_n)$ iff $a_i\ R^{A_i}_{\Gamma\,|\,\Xi}\ f_i$ for $1\le i\le n$, where $\Gamma = x_1 : A_1,\dots x_n : A_n$. Similarly, we can extend q and u to type environments as well.

Proposition 6.2 (Logical Relations Lemma) *If* $\Gamma \vdash e : C$ *and* $\vec{a}\ R^\Gamma_{\Gamma\,|\,\Xi}\ \vec{f}$ *then*

$$[\![e]\!](\vec{a})\ R^C_{\Gamma\,|\,\Xi}\ e[\vec{f}/\vec{x}],$$

where $\vec{x}$ *are the variables in* Γ.

Theorem 6.3 *The equational theory BiCCC is decidable.*

Proof. The above shows that the normalisation function nf satisfies NF1, because by (ii) and $\mathsf{d}(1_\Gamma) = 1_\Gamma$, we know that

$$\mathsf{u}^\Gamma_\Gamma(1_\Gamma)R^\Gamma_\Gamma 1_\Gamma$$

Hence by Proposition 6.2, we know that

$$[\![e]\!](\mathsf{u}^\Gamma_\Gamma(1_\Gamma))R^A_\Gamma e$$

Hence, by (i) (cf. Remark 4.3)

$$\Gamma \vdash \mathsf{d}(\mathsf{nf}(e)) = \mathsf{d}(\mathsf{q}^A_\Gamma([\![e]\!](\mathsf{u}^\Gamma_\Gamma(1_\Gamma)))) = e$$

As we pointed out in the introduction NF2 holds automatically, and hence it follows that

$$\Gamma \vdash e_1 = e_2 \iff \mathsf{nf}(e_1) = \mathsf{nf}(e_2)$$

This yields a decision procedure since equality of normal forms is decidable. (Note that when writing the algorithm we represent the finite set of guards as a list or a tree, so that normal forms are only unique up to the ordering of the guards.) Furthermore, the interpretation in $Sh(\mathcal{N})$ as well as the definition of q,u are clearly algorithmic. In fact, the whole development can be formalised in extensional Martin-Löf type theory using standard methods for formalizing category theory in Martin-Löf type theory. This would be one way of demonstrating explicitly that all functions we construct by abstract mathematical means are computable. $\square$

References

[AHS95] T. Altenkirch, M. Hofmann, T. Streicher, Categorical reconstruction of a reduction-free normalisation proof, *Proc. CTCS '95 Springer LNCS* **953**, 182–199.

[AHS96] T. Altenkirch, M. Hofmann, T. Streicher, Reduction- free normalisation for a polymorphic system, *11th Annual IEEE LICS Symposium*, 1996, 98-106.

[AC98] R. M. Amadio, P-L. Curien, *Selected Domains and Lambda Calculi*, Camb. Univ. Press, 1998.

[BBSSZ98] H. Benl, U. Berger, H. Schwichtenberg, M. Seisenberger and W. Zuber, Proof theory at work: Program development in the Minlog system, in: *Automated Deduction*, W. Bibel and P.H. Schmitt, eds., Vol. II, Kluwer 1998).

[BES98] U. Berger, M. Eberl, H. Schwichtenberg Normalization by evaluation, Prospects for hardware foundations (NADA), *Springer LNCS* **1546**, 1998, pp. 117–137.

[BS91] U. Berger and H. Schwichtenberg, An inverse to the evaluation functional for typed λ- calculus,*6th Annual IEEE LICS Symposium*, 1991, 203-211.

[CD97] T. Coquand and P. Dybjer, Intuitionistic Model Constructions and Normalization Proofs, *Math. Structures in Compter Science* **7**, 1997, 75-94.

[CDS97] D. Čubrić, P. Dybjer and P.J.Scott. Normalization and the Yoneda Embedding, *Math. Structures in Compter Science* **8**, No.2, 1997, 153-192.

[Da96] O. Danvy. Type-directed partial evaluation, POPL'96, ACM Press, 242-257.

[Da98] O. Danvy. Type-directed partial evaluation, Partial evaluation, Practice and Theory, *Proceedings of the 1998 DIKU Summer School*, *Springer LNCS* **1706**, 367-411.

[DiCo95] R. Di Cosmo. *Isomorphism of Types: from λ-calculus to information retrieval and language design*, Birkhäuser, 1995.

[Do93] D. Dougherty. Some lambda calculi with catgorical sums and products, RTA5, *Springer LNCS* **690**, 1993, 135-151.

[DS95] D. Dougherty and R. Subrahmanyam, Equality between Functionals in the Presence of Coproducts, *Inf. & Comp.* (to appear). Prelim. version in: *10th Annual IEEE LICS Symposium*, 1995, 282-291.

[Fil01] A. Filinski, Normalization by Evaluation for the Computational Lambda-Calculus, to appear in the proceedings of TLCA 2001.

[FS99] M. Fiore and A. Simpson. Lambda Definability with Sums via Grothendieck Logical Relations, TLCA'99, *Springer LNCS* **1581**.

[Gh95a] N. Ghani, *Adjoint Rewriting*, PhD thesis, LFCS, Univ. of Edinburgh, Nov. 1995.

[Gh95b] N. Ghani, $\beta\eta$-equality for coproducts. *TLCA '95 Springer LNCS* **902**,1995,171-185.

[GLT89] J.-Y. Girard, Y. Lafont, P. Taylor. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7, 1989.

[JGh95] C. B. Jay and N. Ghani, The virtues of eta expansion, *Journal of Functional Programing*, **5**, no.2, 1995,135-154.

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

[MM92] S. Mac Lane, I. Moerdijk. *Sheaves in Geometry and Logic*, Springer-Verlag, 1992.

[ML75] P. Martin-Löf. An Intuitionistic Theory of Types: Predicative Part, *Logic Colloquium '73*, North-Holland, 1975.

[Mit96] J. C. Mitchell. *Foundations for Programming Languages*, MIT Press, 1996.