Reduction-free normalisation for system F

Thorsten Altenkirch

Institut für Informatik,LMU-München

Oettingenstr. 67

D-80538 München

Germany

e-mail:alti@informatik.uni-muenchen.de

Martin Hofmann and Thomas Streicher

TH Darmstadt, FB 4,

Schloßgartenstr. 7,

64289 Darmstadt, Germany

e-mail:{mh|streicher}@mathematik.th-darmstadt.de

July 2, 1996

Running title:

Reduction-free normalisation for system F

Address for correspondence:

Martin Hofmann

TH Darmstadt, FB 4

Schloßgartenstr. 7

64289 Darmstadt, Germany

e-mail:mh@mathematik.th-darmstadt.de

Abstract

We present a semantical proof of existence of normal forms for system F including η-equality. A reduction-free normalisation function can be obtained from this. The proof uses the method of glueing (a variant of) the term model along the global sections functor, carried out in the internal language of a category of presheaves. As a by-product we obtain an semantical explanation of higher-order abstract syntax. The paper extends a previous one (Altenkirch, Hofmann, and Streicher 1996) in which a combinatory version of system F has been treated.

1 Introduction

In this paper we give a *semantical* proof of *reduction-free normalisation* for $F_{\beta\eta}$, a version of Girard's system F with full $\beta\eta$-equality for both kinds of abstraction. This generalises the semantical normalisation algorithms for simply-typed systems (Berger and Schwichtenberg 1991; Coquand and Dybjer 1996; Altenkirch, Hofmann, and Streicher 1995) to polymorphism.

As in those approaches we do not prove strong normalisation but construct a function nf sending terms to terms in normal form such that convertible terms are sent to the *same* normal form and any term t is convertible with $nf(t)$. Such a function is sufficient for practical purposes as it allows one for every term to compute a normal form and consequently to decide equality of terms. These "normal forms" are computed by structural induction; no notion of term rewriting whatsoever is used although the normal forms computed by our algorithm are also normal forms in the sense of term rewriting, more precisely, they are long $\beta\eta$-normal forms in the sense of e.g. (Huet 1976)

This work is part of a larger programme aiming at deriving reduction-free normalisation algorithms for more complex systems such as Martin-Löf type theory, Extended Calculus of Constructions (ECC), and variants of the Logical Framework. The ultimate goal would be to derive implementations of these systems which would be more efficient than the existing ones because the reduction-free normalisation algorithms can employ the interpreter of the underlying functional programming language as e.g. Standard ML. This gain of efficiency was the initial motivation of Berger and Schwichtenberg for studying reduction-free normalisation.

The case of simply-typed lambda calculus has been treated by the abovementioned authors. The richer systems like ECC extend the simply-typed lambda calculus by two new features: type dependency and polymorphism. We treat the latter in this paper and leave type dependency for future work.

The key idea of the present work (and also implicit in (Coquand and Dybjer 1996)) is to construct a *model* $\mathcal{GT}$ (for Girard-Tait or Glueing of the Term model) in which types are interpreted as quadruples $(A, \mathbf{A}^{pred}, \mathtt{q}^{\mathbf{A}}, \mathtt{u}^{\mathbf{A}})$ where A is a syntactic type, $\mathbf{A}^{pred}$ is a family of "sets" indexed by conversion classes of terms of A, $\mathtt{q}^{\mathbf{A}}$ ("quote") is a function mapping an element of $\mathbf{A}^{pred}([t])$ to a normal form in $[t]$, i.e. convertible with t, and finally $\mathtt{u}^{\mathbf{A}}$ ("unquote") is a function mapping a *neutral term* t, i.e. a variable applied to normal forms, to an element of the set $\mathbf{A}^{pred}([t])$. Unlike in (Coquand and Dybjer 1996), where these "sets" are ordinary sets, in our model they are replaced by presheaves over the finite-product category $\mathbb{V}$ of types and variable renamings which has as objects

pairs $[n, \Gamma]$ where Γ is a $F_{\beta\eta}$-context in n free type variables. A $\mathbb{V}$-morphism from $[n, \Gamma]$ to $[m, \Delta]$ is an m-tuple $\vec{A}$ of types in n free type variables together with a renaming w from Γ to $\Delta[\vec{A}]$.

We summarise our main technical results. Our constructions are carried out within (models of) an impredicative Type Theory, i.e. an extensional version of the Calculus of Constructions extended by inductive types (e.g. similar to (Luo 1994)). We prove that the category $\hat{\mathbb{V}}$ of presheaves over $\mathbb{V}$ with values in the constructive sets of our metalanguage forms itself a model of impredicative type theory. Generalising a result in (Asperti and Martini 1992) we construct a term model of $F_{\beta\eta}$ living inside $\hat{\mathbb{V}}$ in which type quantification becomes "set-theoretic" dependent product (inside $\hat{\mathbb{V}}$). Furthermore, we show that in the term model there exists a higher-order abstraction operation which constructs a term of type $A \Rightarrow B$ from a $\hat{\mathbb{V}}$-function mapping variables of type A to terms of type B. In this way our term model can be seen as a model for *higher-order abstract syntax* in the sense of (Despeyroux, Felty, and Hirschowitz 1995).

The main result consists of the construction of the glued model $\mathcal{GT}$ now internal to the category $Fam(\hat{\mathbb{V}})$ of families or *deliverables* in $\hat{\mathbb{V}}$. The objects of $\mathcal{GT}$ can be considered as a naive version of Tait's computability predicates. Morphisms are maps respecting these. The desired result on reduction-free normalisation is an immediate corollary of the correctness of this model.

Our approach is modular in the sense that we first identify a collection of objects and functions in $\hat{\mathbb{V}}$ together with equations satisfied by those and then construct the model $\mathcal{GT}$ entirely within the internal language of $\hat{\mathbb{V}}$, i.e. using impredicative type theory augmented by these objects and functions (and the equations). Apart from simplifying the exposition this makes it possible to use an implementation of impredicative type theory, for instance *Lego* (Luo and Pollack 1992) to define the model $\mathcal{GT}$ and to prove its correctness. In our previous paper (Altenkirch, Hofmann, and Streicher 1995) we were not yet able to formulate the glued model entirely within the internal language resulting in more complicated definitions and proofs. The methods developed in this paper apply just as well to the simply-typed case dealt with in loc. cit. and would result in a considerable simplification.

We assume some knowledge of category theory and in particular a certain acquaintance with categories of presheaves. A type-theoretically skilled reader who does not have this background may still take profit of our paper by taking the context of definitions summarised in Fig. 8.3 for granted and going through the construction of the model $\mathcal{GT}$ on top of this context.

2 Syntax

The system $F_{\beta\eta}$ is a variant of Girard's System F with full $\beta\eta$-equality. We use de Bruijn indices to represent variables; the judgement $n \vdash A$ means that A is a type with at most n free variables and is defined as follows :

$$\frac{i < n}{n \vdash i}\ (\textsc{Var}) \qquad \frac{n \vdash A \qquad n \vdash B}{n \vdash A \Rightarrow B}\ (\Rightarrow) \qquad \frac{n+1 \vdash A}{n \vdash \forall(A)}\ (\forall)$$

We define the category $\mathbb{S}$ of type substitutions : objects are natural numbers and a morphism in $\mathbb{S}(m,n)$ is an n-tuple of types $\vec{A} = (A_{n-1}, \ldots A_0)$ where $m \vdash A_i$ for $i = 0, \ldots n-1$. Notice the right to left decomposition of such tuples, i.e. the "first" element of $\vec{A}$ is A_0, its leftmost entry.

Composition is given by substitution which can be defined as a recursive function over the syntax. $\mathbb{S}$ has a terminal object 0 and finite products which are given on objects by addition of natural numbers. Here and in the sequel we use the standard notation π, π', $\langle -, - \rangle$ for projections and pairing possible decorated with typing subscripts. We write $\langle\rangle$ for the unique morphism into a terminal object.

If $n+1 \vdash A$ and $n \vdash B$ then we write $A[B]$ for $A \circ \langle id_n, B \rangle$ and B^+ for $B \circ \pi_{n,1}$. We have $n \vdash A[B]$ and $n+1 \vdash B^+$.

Given a natural number n, a context $n \vdash \Gamma$ is a sequence of types in n variables, i.e. of types A such that $n \vdash A$. We denote the length of a context by $|\Gamma|$ and decompose Γ as $\Gamma = (\Gamma_{|\Gamma|-1}, \ldots \Gamma_0)$. Note that a context $n \vdash \Gamma$ is an $\mathbb{S}$-morphism from n to $|\Gamma|$. We write $\diamond$ for the empty context and $\Gamma.A$ for the extension of context Γ by type A. The operation Γ^+ arises as the pointwise extension of A^+. We define a judgement $\Gamma \vdash_n t : A$, where $n \vdash \Gamma$ and $n \vdash A$ by the following rules:

$$\frac{i < |\Gamma|}{\Gamma \vdash_n i_\Gamma : \Gamma_i}\ (\textsc{tm-var})$$

$$\frac{\Gamma.A \vdash_n t : B}{\Gamma \vdash_n \lambda_{\Gamma,A,B}(t) : A \to B}\ (\lambda) \qquad \frac{\Gamma \vdash_n t : A \to B \qquad \Gamma \vdash_n s : A}{\Gamma \vdash_n \mathrm{app}_{\Gamma,A,B}(t,s) : B}\ (\textsc{app})$$

$$\frac{\Gamma^+ \vdash_{n+1} t : A}{\Gamma \vdash_n \Lambda_{\Gamma,A}(t) : \forall(A)}\ (\Lambda) \qquad \frac{\Gamma \vdash_n t : \forall(A) \qquad n \vdash B}{\Gamma \vdash_n \mathrm{App}_{\Gamma,A}(t,B) : A[B]}\ (\textsc{App})$$

The type annotations simplify the interpretation of the syntax. We may drop the annotations when they are clear from the context and write ts for $\mathrm{app}_{\Gamma,A,B}(t,s)$ and tB for $\mathrm{App}_{\Gamma,A}(t,B)$.

For any n we define a category of terms $\mathbb{T}^0_n$ whose objects are contexts $n \vdash \Gamma$ and morphisms between Γ and Δ are context morphisms, i.e. sequences of terms $\Gamma \vdash t_i : \Delta_i$. Composition of context

morphisms is defined as usual as simultaneous substitution which can be defined by structural induction over the syntax. The categories $\mathbb{T}^0_n$ have products given on objects by juxtaposition of contexts. As in the case of the category of type substitutions we introduce the special syntax $t[s]$ for substitution of a single term into another term ($t[s] = t \circ \langle id, s \rangle$) and s^{+X} for weakening with the type X, i.e. $s^{+X} = s \circ \pi_{\Gamma,X}$. If $\Gamma.A \vdash_n t : B$ and $\Gamma \vdash s : A$ then $\Gamma \vdash_n t[s] : B$ and $\Gamma.X \vdash s^{+X} : A$. We also extend type substitution and weakening to terms in the obvious way, i.e. if $\Gamma \vdash_{n+1} t : A$ and $n \vdash B$ then $\Gamma[B] \vdash_n t[B] : A[B]$ and if $\Gamma \vdash_n t : A$ then $\Gamma^+ \vdash_{n+1} t^+ : B^+$. More generally, if $\vec{A} \in \mathbb{S}(m,n)$ and $\Gamma \vdash_n t : A$ then we have $\Gamma[\vec{A}] \vdash_m t[\vec{A}] : A[\vec{A}]$. In this way we obtain an indexed category $\mathbb{T}^0 : \mathbb{S}^{\mathrm{op}} \to \mathbf{Cat}$ (with fibres $\mathbb{T}^0_n$). We also write $\mathbb{T}^0$ for the total category of the associated split fibration, i.e. the category $\mathbb{T}^0$ has as objects pairs $[n, \Gamma]$ where $n \vdash \Gamma$ and a morphism from $[n, \Gamma]$ to $[m, \Delta]$ is a pair $[\vec{A}, \vec{t}]$ such that $\vec{A} \in \mathbb{S}(n,m)$ and $\vec{t} \in \mathbb{T}^0_n(\Gamma, \Delta[\vec{A}])$.

2.1 Equational theory

For t, s such that $\Gamma \vdash_n t, s : A$ we define a conversion relation $conv_{\Gamma,A}$ by the least congruence containing the following equations :

$$\begin{array}{rll}
\mathrm{app}_{\Gamma,A,B}(\lambda_{\Gamma,A,B}(t), s) & conv_{\Gamma,B} & t[s] \\
\lambda_{\Gamma,A,B}(\mathrm{app}_{\Gamma,A,B}(t^{+A}, 0_{\Gamma.A})) & conv_{\Gamma,A\to B} & t \\
\mathrm{App}_{\Gamma,A}(\Lambda_{\Gamma,A}(t), B) & conv_{\Gamma,A[B]} & t[B] \\
\Lambda_{\Gamma,A}(\mathrm{App}_{\Gamma,A}(t^+, 0)) & conv_{\Gamma,\forall(A)} & t
\end{array}$$

For $n \in \mathbb{N}$ we define the category $\mathbb{T}_n$ as the quotient of $\mathbb{T}^0_n$ by $conv$. The objects of $\mathbb{T}_n$ are contexts $n \vdash \Gamma$ and the morphisms are $conv$-equivalence classes of $\mathbb{T}^0_n$-morphisms, i.e. tuples of terms. Since $conv$ is a congruence it is respected by weakening and substitution so that these operations lift to equivalence classes. We identify notationally classes with representatives. In this way, we obtain another indexed category $\mathbb{T} : \mathbb{S}^{\mathrm{op}} \to \mathbf{Cat}$ together with the associated split fibration $\mathbb{T} \to \mathbb{S}$.

2.2 The category of variable renamings

For $n \vdash \Gamma$ and $n \vdash \Delta$ we define $\mathbb{V}_n(\Gamma, \Delta)$ as the set of those morphisms in $\mathbb{T}_n(\Gamma, \Delta)$ which consist of variables (natural numbers) only, i.e. do not contain the term formers app, λ, etc. Clearly, these morphisms are closed under composition and contain the identities so that we have an indexed

sub-category $\mathbb{V}$ of $\mathbb{T}$ and $\mathbb{T}^0$. The composition of the inclusion from $\mathbb{V}$ into $\mathbb{T}^0$ with the projection from $\mathbb{T}^0$ to $\mathbb{T}$ furnishes a product-preserving functor $\mathcal{I}$ from $\mathbb{V}$ to $\mathbb{T}$. We do not assume that $\mathcal{I}$ is monic although this follows a posteriori from our normalisation argument. Again, $\mathbb{V}$ also denotes the total category of the associated split fibration. Objects of $\mathbb{V}$ are pairs $[n, \Gamma]$ where $n \vdash \Gamma$ and morphisms between $[n, \Gamma]$ and $[m, \Delta]$ are pairs $[\vec{A}, w]$ with $\vec{A} \in \mathbb{S}(n, m)$ and $w \in \mathbb{V}_n(\Gamma, \Delta[\vec{A}])$.

Notice that the total categories $\mathbb{V}$, $\mathbb{T}^0$, and $\mathbb{T}$ each have products by "abstract nonsense" since the total category of a fibration has finite products if the base and each fibre has finite products and reindexing preserves them. For example, the product of $[n, \Gamma]$ and $[1, \diamond]$ is given by $[n+1, \Gamma^+]$ with projections $[\pi_{n,1}, id_\Gamma]$ and $[\pi'_{n,1}, \langle\rangle]$.

Our glued model will be internal to the presheaf category $\hat{\mathbb{V}}$. The argument to be made would also go through with the more economic sub-category of *weakenings* (used in (Altenkirch, Hofmann, and Streicher 1995)) which consists of order-preserving variable renamings, i.e. the vectors of the form $(i_{m-1}, \ldots i_0)$ where $i_{m-1} \geq i_{m-2} \geq \cdots \geq i_0$. Using this category would, however, make the exposition more complicated, in particular in Section 8.2, so that we have opted for $\mathbb{V}$. If an efficient algorithm is to be extracted from our development then the category of weakenings (or even the category of projections to initial segments, which also works albeit with a yet more complex development) should be used instead.

2.3 Normal forms and neutral terms

The sets NF and NE of *normal forms* and *neutral terms*, respectively, are given by the following grammar (where $t \in NF$ and $u \in NE$ and $i \in \mathbb{N}$):

$$\begin{array}{lcl} NF & ::= & u \mid \lambda_{\Gamma,A,B}(t) \mid \Lambda_{\Gamma,A}(t) \\ NE & ::= & i \mid \mathrm{app}_{\Gamma,A,B}(u, t) \mid \mathrm{App}_{\Gamma,A}(u, B) \end{array}$$

Notice that normal forms and neutral terms are stable under type substitution and variable renamings in the sense that if $[\vec{A}, w] \in \mathbb{V}([n, \Gamma], [m, \Delta])$ and $t \in NF$ satisfies $\Delta \vdash_m t : A$ then the term $\Gamma \vdash_n t[\vec{A}][w] : A[\vec{A}]$ is in normal form, too. The same goes for neutral terms. There can be more than one normal form in each conversion class; for an example see Section 8.3. The reason is that a normal form of universally quantified type can either be neutral (for instance a variable) or a Λ-abstraction.

3 Constructive metalanguage

We understand the previous and the following definitions to be made within a constructive metalanguage which contains an impredicative universe *Prop* of *small sets* closed under inductive definitions. Impredicativity means that the product of an arbitrary family of small sets is again small. Furthermore, we require subset types for equality predicates (i.e. equalisers) and quotients of classical (i.e. $\neg\neg$-closed) equivalence relations. A model for such a metalanguage is furnished by ω-sets and pers/modest sets as described in (Streicher 1991). For a proof that the ω-set model admits the required quotients see for instance Ch. II of loc. cit.

We use an informal (impredicative) extensional Martin-Löf type theory to denote constructions in the metalanguage. In particular, we write Π and Σ for dependent product and sum, and we use λ and juxtaposition for abstraction and application and $\langle -, - \rangle$ and .1, .2 for pairing and projections. We use the symbol *Type* for the universe of sets in the metalanguage and we form kinds like $A \to \mathit{Type}$ to denote the class of families of sets indexed over A. If $B : A \to \mathit{Type}$ and $f(x), g(x) : B(x)$ if $x\colon A$ then we write $\{x\colon A \mid f(x) = g(x)\}$ for the subset of A where f and g agree.

We use the symbols $\in$ and : for membership interchangeably but mostly employ the colon when Martin-Löf type theory is being used.

Many of our constructions are taking place in specific models of our metalanguage. We overload the syntax of the type theoretic operations but may disambiguate by $\mathit{Type}^C, \Pi^C, \Sigma^C$ where C is the model. Since the models are defined inside our constructive metalanguage all internal constructions can be expanded and replaced by much longer (and unreadable) definitions in the metalanguage.

If $a\colon A$ and $f(a) = g(a)$ then we may write $a : \{x\colon A \mid f(x) = g(x)\}$. Conversely, if $a : \{x\colon A \mid f(x) = g(x)\}$ then $a : A$ and furthermore $f(a) = g(a)$ may be inferred. If A is a small set then so is $\{x\colon A \mid f(x) = g(x)\}$.

For the definition of functions we either use λ-notation like in $f = \lambda x\colon A.t$ or equivalently a "pointwise" notation like in $f(x\colon A) = t$. For iterated application we sometimes use parentheses and commas, e.g. if $f : A \to \Pi b\colon B.C(b)$ and $a\colon A$ and $b\colon B$ then we may write $f(a, b)$ instead of $f\, a\, b$. We sometimes omit arguments which can be inferred. For instance, if $f : \Pi a\colon A.B(a) \to C$ and $a\colon A$ and $b\colon B(a)$ then we may abbreviate $f(a, b)$ by $f(b)$.

4 Naive F-doctrines

Definition 1 *A contextual cartesian closed category (ConCCC) is a category $\mathcal{C}$ together with a distinguished sub-class $Ty = Ty^{\mathcal{C}}$ such that*

- *$\mathcal{C}$ has a terminal object,*
- *$\mathcal{C}$ has all cartesian products of the form $\Gamma \times A$ where $\Gamma \in \mathcal{C}$ and $A \in Ty$,*
- *$\mathcal{C}$ has all exponentials of the form $A \Rightarrow B$ for $A, B \in Ty$*

Every cartesian-closed category $\mathcal{C}$ is a ConCCC with the setting $Ty = |\mathcal{C}|$. The advantage of ConCCC's over cartesian-closed categories is that term models furnish more naturally a ConCCC.

Proposition 2 *The indexed category $\mathbb{T} : \mathbb{S}^{\mathrm{op}} \to$ **Cat** from Section 2 is an indexed ConCCC, i.e. $\mathbb{T} : \mathbb{S}^{\mathrm{op}} \to$ **ConCCC** where **ConCCC** is the category of ConCCC's and structure-preserving functors. Moreover, the functors $\mathbb{T}(\pi_{n+1,n}) : \mathbb{T}_n \to \mathbb{T}_{n+1}$ have right adjoints satisfying the Beck-Chevalley condition up to equality, i.e. $\mathbb{T}$ admits split Cons_1-products in the sense of Jacobs (1991, §1.5). Moreover, $0 \in \mathbb{T}_1$ is a generic object for $\mathbb{T}$.*

Proof. Standard. See, e.g. (Crole 1993). For the ease of the reader we give some of the raw data here. The cartesian product of Γ and A in $\mathbb{T}_n$ is given by the context $\Gamma.A$. The exponential of B by A in $\mathbb{T}_n$ is given by the type $A \Rightarrow B$. Finally, the result of applying the right adjoint of $\mathbb{T}(\pi_{n+1,n})$ to $A \in \mathbb{T}_{n+1}$ is the type $\forall(A) \in \mathbb{T}_n$. If $t \in \mathbb{T}_{n+1}(\Gamma^+, A)$ then its transpose is the term $\Lambda_{\Gamma,A}(t) \in \mathbb{T}_n(\Gamma, \forall(A))$.

Notice that we have $|\mathbb{T}_n| = \mathbb{S}(n, 1)$ which establishes that $0 \in \mathbb{T}_1$ (recall that 0 refers to the single type variable corresponding to the judgement $\vdash_1 0$) is a generic object. □

For further reference we call such a structure an *external model*. We remark that up to the distinction between contexts and types $\mathbb{T}$ is a model of system F in the sense of Seely (Crole 1993). For the subsequent development it is, however, more appropriate to employ a different, but equivalent notion of model due to Asperti and Longo (1992) based on internal categories. Their notion is adapted to the ConCCC-situation in the following definition where the model of our metalanguage plays the role of their ambient category.

Definition 3 *A naive F-doctrine is a ConCCC $\mathcal{C}$ which has Ty-indexed products of types, i.e. for every function $B : Ty \to Ty$ the product of B in $\mathcal{C}$ exists and lies in Ty.*

Let $\mathcal{C}$ be a naive F-doctrine. In the following we introduce some notation and names to refer to its components. The objects of $\mathcal{C}$ are called contexts, the objects in Ty are called types. We write $\diamond$ for the terminal object and $\langle\rangle_\Gamma \in \mathcal{C}(\Gamma, \diamond)$ for the unique morphism into $\diamond$. If Γ is a context and A is a type then we write $\Gamma.A$ for their cartesian product and $\pi_{\Gamma,A} \in \mathcal{C}(\Gamma.A, \Gamma)$ and $\pi'_{\Gamma,A} \in \mathcal{C}(\Gamma.A, A)$. If $\Delta = \diamond.A_1.\ldots.A_k$ for types A_i then the cartesian product of Γ and Δ also exists in $\mathcal{C}$ and we write it as $\Gamma.\Delta$ and use the already introduced notation for the associated morphisms with the A-subscript replaced by Δ. We (partially) omit these and subsequent subscripts wherever appropriate.

If A and B are types then we write $A \Rightarrow B$ for their exponential in $\mathcal{C}$. If $f \in \mathcal{C}(\Gamma, A \Rightarrow B)$ and $g \in \mathcal{C}(\Gamma, A)$ then we write $\mathrm{app}_{\Gamma,A,B}(f,g) \in \mathcal{C}(\Gamma, B)$ for the composition of $\langle f, g\rangle$ with the evaluation morphism. If $f \in \mathcal{C}(\Gamma.A, B)$ then we write $\lambda_{\Gamma,A,B}(f) \in \mathcal{C}(\Gamma, A \Rightarrow B)$ for the exponential transpose of f.

For $B : Ty \to Ty$ we write $\forall(B)\colon Ty$ for the product of B and for every $X\colon Ty$ we write $\mathrm{App}_B(X)\colon \mathcal{C}(\forall(B), B(X))$ for the product projection corresponding to X. If $f : \Pi X\colon Ty.\mathcal{C}(\Gamma, B(X))$ is a Ty-indexed family of morphisms then $\Lambda_B(f) : \mathcal{C}(\Gamma, \forall(B))$ denotes the (unique) morphism such that for all $X\colon Ty$

$$\mathrm{App}_B(X) \circ \Lambda_B(f) = f(X) \qquad \text{Ty-Beta}$$

Whenever $f\colon \mathcal{C}(\Gamma, \forall(B))$ then we have

$$\Lambda_B(\lambda X\colon Ty.\mathrm{App}_B(X) \circ f) = f \qquad \text{Ty-Eta}$$

We decorate these symbols with superscripts $^{\mathcal{C}}$ identifying the corresponding naive F-doctrine wherever this appears appropriate. Instead of $\mathcal{C}(A, B)$ we may write $Mor(A, B)$ or $Mor^{\mathcal{C}}(A, B)$ for the set of morphisms.

Example 4 *An example of a naive F-doctrine is furnished by the category $\mathcal{P}$ which has as objects the small sets (elements of Prop) and as morphisms the functions between them. We put $Ty^{\mathcal{P}} = |\mathcal{P}| = Prop$. It is clear that this defines a ConCCC. As for type quantification we use the impredicativity of Prop, i.e. for $B : Prop \to Prop$ we define $\forall(B)$ as $\Pi X\colon Prop.B(X)$.*

5 Interpretation of the syntax

Fix a naive F-doctrine $\mathcal{C}$. By induction over the syntax we define a many-sorted interpretation function $[\![-]\!] = [\![-]\!]^{\mathcal{C}}$ which maps a context Γ in n free type variables to a function $[\![n \vdash \Gamma]\!]$ from

$$
\begin{array}{lcl}
[\![n \vdash \diamond]\!](\vec{X}{:}(Ty^{\mathcal{C}})^n) & = & \diamond \\
[\![n \vdash \Gamma.A]\!](\vec{X}) & = & [\![n \vdash \Gamma]\!](\vec{X}).[\![n \vdash A]\!](\vec{X}) \\
[\![n \vdash i]\!](\vec{X}) & = & X_i \qquad \text{where } \vec{X} = (X_{n-1}, \ldots X_0) \\
[\![n \vdash A \to B]\!](\vec{X}) & = & [\![n \vdash A]\!](\vec{X}) \Rightarrow [\![n \vdash B]\!](\vec{X}) \\
[\![n \vdash \forall(A)]\!](\vec{X}) & = & \forall(\lambda X{:}\, Ty^{\mathcal{C}}.[\![n+1 \vdash A]\!](\langle \vec{X}, X \rangle)) \\
 & & \\
[\![\Gamma \vdash_n i_\Gamma : \Gamma_i]\!](\vec{X}) & = & \text{projection from } [\![n \vdash \Gamma]\!](\vec{X}) \text{ to } [\![n \vdash \Gamma_i]\!] \\
[\![\Gamma \vdash_n \lambda_{\Gamma,A,B}(t)]\!](\vec{X}) & = & \lambda([\![\Gamma.A \vdash_n t : B]\!](\vec{X}) \\
[\![\Gamma \vdash_n \mathrm{app}_{\Gamma,A,B}(t,s)]\!](\vec{X}) & = & \mathrm{app}([\![\Gamma \vdash_n t : A]\!](\vec{X}), [\![\Gamma \vdash_n s : B]\!](\vec{X})) \\
[\![\Gamma \vdash_n \Lambda_{\Gamma,A}(t) : \forall(A)]\!](\vec{X}) & = & \Lambda(\lambda X{:}\, Ty^{\mathcal{C}}.[\![\Gamma^+ \vdash_{n+1} t : A]\!](\langle \vec{X}, X \rangle)) \\
[\![\Gamma \vdash_n \mathrm{App}_{\Gamma,A}(t,B) : A[B]]\!](\vec{X}) & = & \mathrm{App}([\![\Gamma \vdash_n t : \forall(A)]\!](\vec{X}), [\![n \vdash B]\!](\vec{X}))
\end{array}
$$

Figure 1: Interpretation of $F_{\beta\eta}$

$(Ty^{\mathcal{C}})^n$ to $|\mathcal{C}|$ and a type A in n free variables to a function $[\![n \vdash A]\!]$ from $(Ty^{\mathcal{C}})^n$ to $Ty^{\mathcal{C}}$. A term t in n free type variables and in context Γ of type A is mapped to a (dependent) function $[\![\Gamma \vdash_n t : A]\!]$ of type

$$\Pi\vec{X}{:}\,(Ty^{\mathcal{C}})^n.\mathbb{T}_n([\![n \vdash \Gamma]\!](\vec{X}), [\![n \vdash A]\!](\vec{X}))$$

The semantic equations are straightforward: syntactic constructs are interpreted by their semantic counterparts. The details are given in Fig. 5.

Theorem 5 (Soundness) *The interpretation is sound, i.e. the interpretation functions are total in the domains described above and conversion is reflected by equality in the model: If t $conv_{\Gamma,A}$ s then $[\![\Gamma \vdash_n t : A]\!] = [\![\Gamma \vdash_n t : A]\!]$.*

Proof. By induction on derivations. □

6 Presheaves as a model of the constructive metalanguage

The notions of ConCCC and naive F-doctrine make sense in any model of the constructive metalanguage. Of particular interest are categories of presheaves $\hat{\mathbf{K}}$ and categories of the form $Fam(\hat{\mathbf{K}})$ of

families of presheaves over $\mathbf{K}$. We explain how these categories model the constructive metalanguage and introduce some notation.

6.1 Categories of presheaves

Let $\mathbf{K}$ be a small category. It is well-known that the category $\hat{\mathbf{K}} = \mathbf{Set}^{\mathbf{K}^{\mathrm{op}}}$ of presheaves supports extensional Martin-Löf type theory together with inductive definitions even if the ambient set-theoretic universe with respect to which $\hat{\mathbf{K}}$ is formed is not a topos, but only a model of Martin-Löf type theory itself. We do not require quotient types on the level of presheaf categories. We refer to (Phoa 1992) for the precise definition of the interpretation of Martin-Löf type theory in a category of presheaves and only sketch some important aspects here. Dependent types in $\hat{\mathbf{K}}$ can be understood via the equivalence of categories (Moerdijk and Lane 1992, p. 157)

$$\hat{\mathbf{K}}/F \cong \widehat{El(F)}$$

where $El(F) = \mathbf{y} \downarrow F$ is the category of elements of the presheaf F. It has as objects pairs (X, f) where $f \in F_X$ and a morphism from (X, f) to (X', f') is a $\mathbf{K}$-morphism $u : X \to X'$ such that $F_u(f') = f$. (We use subscripts for application of presheaves.) We write $\hat{\mathbf{K}}(F)$ for $\widehat{El(F)}$. So for example the objects of kind $F \to \mathit{Type}^{\hat{\mathbf{K}}}$ are the objects of $\hat{\mathbf{K}}(F)$. Cartesian products and Σ-types are taken pointwise: $(U \times V)_I = U_I \times V_I$ and $(\Sigma u{:}U.V(u))_I = \Sigma u{:}U(I).V_I(u)$. The same goes for subset types.

If $U \in \hat{\mathbf{K}}(F)$ we write $F_I(u)$ (instead of $F_{(I,u)}$) for the application of F to the pair (I, u). We sometimes call I a "stage" or a "world".

The function space of $U, V \in \hat{\mathbf{K}}$ is given by the presheaf which sends $I \in \mathbf{K}$ to the set $\hat{\mathbf{K}}(\mathbf{y}(I) \times U, V)$, where $\mathbf{y} : \mathbf{K} \to \hat{\mathbf{K}}$ is the Yoneda embedding. More elementarily, the elements of $(U \to V)_I$ are mappings

$$\mu \in \Pi J{:}\mathbf{K}.\mathbf{K}(J, I) \times U_J \to V_J$$

such that whenever $u \in \mathbf{K}(J, I)$, $u' \in \mathbf{K}(J', J)$, and $f \in U_J$ then $V_{u'}(\mu_J(u, f)) = \mu_{J'}(u \circ u', U_{u'}(f))$.

We remark that by "uncurrying" a global element of a presheaf exponential, i.e. a natural transformation $1 \longrightarrow (U \to V)$ is nothing but a natural transformation from U to V. More generally, since $U \to (V \to W)$ is isomorphic to $(U \times V) \to V$ a global element of $U \to (V \to W)$ can be given by specifying a natural transformation from $U \times V$ to W. The same goes for longer chains of arrows.

This generalises to "non-empty contexts" in view of the definition of $\hat{\mathbf{K}}(F)$ as a category of presheaves, i.e. the local exponential of $G, H \in \hat{\mathbf{K}}(F)$ can be obtained by replacing $\mathbf{K}$ by $El(\mathbf{K})$ in the definition of ordinary exponentials. We finally remark that dependent products can be obtained as regular subsets of suitable exponentials, e.g. if $U \in \hat{\mathbf{K}}$ and $V \in \hat{\mathbf{K}}(U)$ then the dependent product $\Pi u{:}\, U.V(u) \in \hat{\mathbf{K}}$ equals $\{f : U \to \Sigma u{:}\, U.V(u) \mid -.1 \circ f = id_U\}$.

This means that global elements of dependent products can also be given element-wise.

6.2 Impredicative universe

Less well-known is the existence of a small impredicative universe in $\hat{\mathbf{K}}$. Call a presheaf $F \in \hat{\mathbf{K}}$ "small" if for every $I \in \mathbf{K}$ the set F_I is small, i.e. lies in *Prop*. More generally, if G is any presheaf, we can consider the set $Small(G)$ of small presheaves in $\hat{\mathbf{K}}(G)$. An element of $Small(G)$ is a functor from the opposite of the category $El(G)$ of elements of G to the category of small sets. The operation $Small(-)$ extends to a functor from $(\hat{\mathbf{K}})^{\mathrm{op}}$ to the category of sets by putting for $\alpha : G' \to G$ and $F \in Small(G)$

$$Small(\alpha)(F)_I(g) = F(I, \alpha_I(g))$$

and if $u : I' \to I$ and thus $u \in El(G')((I', G'_u(g)),\ (I, g))$ and

$$Small(\alpha)(F)_u(f) = F_u(f)$$

which works because $u \in El(G)((I', G_u(\alpha_{X'}(u))),\ (I, \alpha_I(u)))$ by naturality of α. It is obvious that the product of a family of small presheaves is again small because its construction only involves taking products of small sets. The important point is that the small families are representable:

Theorem 6 *The small presheaves over G are in bijective correspondence with $\hat{\mathbf{K}}$-morphisms from G to the presheaf Prop defined by $Prop_I = Small(\mathbf{y}(I))$*

Proof. If $F \in Small(G)$ then the associated morphism $\chi_F : G \to Prop$ at $I \in \mathbf{K}$ sends $g \in G_I$ to the composition $F \circ \bar{g}$ where $\bar{g} : \mathbf{K}/I \to El(G)$ maps $u : J \to I$ to $(J, G_u(g))$.

Conversely, if $f : G \to Prop$ then we obtain $Prf(f) \in Small(G)$ as the presheaf which at $(I, g) \in El(G)$ is $f_I((I, g), id_I)$.

For $F \in Small(G)$ we have $Prf(\chi_F) = F$ by functoriality of G and for $f : G \to Prop$ we have $\chi_{Prf(f)} = f$ by naturality of f. □

We shall henceforth identify small presheaves with the associated morphisms into *Prop* and use the informal Martin-Löf type theory explained above in Sect. 3 also to denote constructions in presheaf categories.

6.3 Interpretation of the syntax in a naive F-doctrine inside $\hat{\mathbf{K}}$

Assume a naive F-doctrine $\mathcal{C}$ inside some category of presheaves $\hat{\mathbf{K}}$. This means that we have a presheaf $|\mathcal{C}| \in \hat{\mathbf{K}}$ (internally $|\mathcal{C}| : \mathit{Type}^{\hat{\mathbf{K}}}$), a sub-presheaf $\mathit{Ty}^{\mathcal{C}} \subset |\mathcal{C}|$, and a presheaf $\mathcal{C} \in \hat{\mathbf{K}}(|\mathcal{C}|, |\mathcal{C}|)$ of morphisms (internally $\mathcal{C} : |\mathcal{C}| \times |\mathcal{C}| \to \mathit{Type}^{\hat{\mathbf{K}}}$), as well as the constants specified after Def. 1 whose types are now understood w.r.t. the internal language of $\hat{\mathbf{K}}$. We remark that this amounts to an *internal model* in the sense of (Asperti and Martini 1992) with ambient category $\hat{\mathbf{K}}$. Following §5 of loc. cit. we define by induction on derivations a semantic function $[\![-]\!] = [\![-]\!]^{\mathcal{C}}$ which to $n \vdash \Gamma$ associates a $\hat{\mathbf{K}}$-morphism from $(\mathit{Ty}^{\mathcal{C}})^n$ to $|\mathcal{C}|$, i.e. we have

$$[\![n \vdash \Gamma]\!] \in \hat{\mathbf{K}}((\mathit{Ty}^{\mathcal{C}})^n, |\mathcal{C}|)$$

Viewed externally, $[\![n \vdash \Gamma]\!]$ assigns to $I \in \mathbf{K}$ and $\vec{X} \in (\mathit{Ty}^{\mathcal{C}}_I)^n$ an element $[\![n \vdash \Gamma]\!]_I(\vec{X}) \in |\mathcal{C}|_I$ natural in I. The same goes for types. A term $\Gamma \vdash_n t : A$ gets interpreted as a global element of the presheaf

$$\Pi \vec{X} \colon (\mathit{Ty}^{\mathcal{C}})^n . \mathcal{C}([\![n \vdash \Gamma]\!](\vec{X}), [\![n \vdash A]\!](\vec{X}))$$

Externally, this is an assignment which to $I \in \mathbf{K}$ and $\vec{X} \in (\mathit{Ty}^{\mathcal{C}}(I))^n$ associates an element

$$[\![\Gamma \vdash_n t : A]\!]_I(\vec{X}) \in \mathcal{C}_I([\![n \vdash \Gamma]\!]_I(\vec{X}), [\![n \vdash A]\!]_I(\vec{X}))$$

again natural in I.As a typical example, we define here the interpretation of the $\forall$-operator. Suppose we are given $[\![n+1 \vdash B]\!] \in \hat{\mathbf{K}}((\mathit{Ty}^{\mathcal{C}})^{n+1}, \mathit{Ty}^{\mathcal{C}})$. The exponential transpose $\mathit{cur}([\![n+1 \vdash B]\!])$ is a morphism in $\hat{\mathbf{K}}$ from $(\mathit{Ty}^{\mathcal{C}})^n$ to the exponential $\mathit{Ty}^{\mathcal{C}} \to \mathit{Ty}^{\mathcal{C}}$. Composition with the semantic $\forall$-operator yields the desired meaning of $[\![n \vdash \forall(B)]\!]$:

$$[\![n \vdash \forall(B)]\!] = \forall \circ \mathit{cur}([\![n+1 \vdash B]\!])$$

As explained above, we can also use λ-notation to define morphisms in $\hat{\mathbf{K}}$. This leads to the more familiar looking definition

$$[\![n \vdash \forall(B)]\!] = \lambda \vec{X} \colon (\mathit{Ty}^{\mathcal{C}})^n . \forall(\lambda X \colon \mathit{Ty}^{\mathcal{C}} . [\![n+1 \vdash B]\!](\vec{X}, X))$$

where the λ-abstraction is interpreted in $\hat{\mathbf{K}}$. In this way we can understand all the other equations from Fig. 5 by reading the right-hand sides as definitions in the internal language of global elements. Again, we have the following soundness property:

Proposition 7 *The interpretation of the syntax in a naive F-doctrine internal to a category of presheaves is sound in the sense that whenever $\Gamma \vdash_n t = s : A$ then $[\![\Gamma \vdash_n t : A]\!] = [\![\Gamma \vdash_n s : A]\!]$.*

We remark that such interpretation function can be defined for naive F-doctrines living inside an arbitrary model $\mathbf{E}$ of the metalanguage provided that the set of $\mathbf{E}$-functions from X to Y in $\mathbf{E}$ is a (constructive) set. We stress that this interpretation function is not directly an instance of the interpretation function defined in 5 above because unlike there syntax and semantics live in different ambient set-theoretic universes.

6.4 Families of presheaves

For a model $\mathbf{E}$ of the constructive metalanguage, for instance $\mathbf{E} = \hat{\mathbf{K}}$, let $Fam(\mathbf{E})$ denote the category whose objects are pairs $A = (A^{syn}, A^{sem})$ where $A^{syn} : Type^{\mathbf{E}}$ and $A^{sem} \in A^{syn} \to Type^{\mathbf{E}}$. A morphism from A to B is a pair $f = (f^{syn}, f^{sem})$ such that $f^{syn}\colon A^{syn} \to B^{syn}$ and $f^{sem} : \Pi a\colon A^{syn}.A^{sem}(a) \to B^{sem}(f^{syn}(a))$. Composition and identities are taken component-wise. We call the first component of an object, respectively a morphism, the *syntactic component*, and use the term *semantic component* for the second one. This terminology is motivated from the fact that in subsequent applications the first components very often are syntactic objects whereas the second components are predicates on syntactic components or proofs of such properties. We have a functor $Fst : Fam(\mathbf{E}) \to \mathbf{E}$ which singles out the syntactic components, i.e. we have $Fst(X) = X^{syn}$ for X an object or a morphism.

Such categories of families can be subsumed under categories of presheaves because of the equivalence

$$Fam(\mathbf{E}) \simeq \mathbf{E}^{\mathbf{2}^{\mathrm{op}}}$$

where $\mathbf{2}$ is the category (in $\mathbf{E}$) with two objects and one non-identity arrow. In the case where $\mathbf{E}$ itself is of the form $\hat{\mathbf{K}}$ we can avoid to consider presheaf-valued presheaves by employing the equivalence

$$Fam(\hat{\mathbf{K}}) \simeq \widehat{\mathbf{K} \times \mathbf{2}}$$

From this we know immediately that families of presheaves furnish a model of the constructive metalanguage. For subsequent use we give the constructions of exponentials, dependent types, and dependent products in $Fam(\mathbf{E})$ in terms of the internal language of $\mathbf{E}$. If A, B are objects in $Fam(\mathbf{E})$ then the exponential is defined by $(A \to B)^{syn} = A^{syn} \to B^{syn}$ and $(A \to B)^{sem}(f\colon (A \to$

$B)^{syn}) = \Pi a{:}\, A^{syn}.A^{sem}(a) \to B^{sem}(f^{syn}(a))$. A type depending on A can be presented as a pair $F = (F^{syn}, F^{sem})$ where $F^{syn} : A \to \mathit{Type}^{\mathbf{E}}$ and $F^{sem} : \Pi a{:}\, A^{syn}.A^{sem}(a) \times F^{syn}(a) \to \mathit{Type}^{\mathbf{E}}$. The dependent product over such a family has syntactic component

$$\Pi(A,F)^{syn} = \Pi a{:}\, A^{syn}.F^{syn}(a)$$

and semantic component

$$\Pi(A,F)^{sem}(f{:}\,\Pi(A,F)^{syn}) = \Pi a{:}\, A^{syn}.\Pi \mathbf{a}{:}\, A^{sem}(a).F^{sem}(a,\mathbf{a},f(a))$$

It is well-known and obvious from these settings that the functor $\mathit{Fst} : \mathit{Fam}(\mathbf{E}) \to \mathbf{E}$ preserves all the structure of a model of the constructive metalanguage.

We remark that $\mathit{Fam}(\mathbf{E})$ can be seen as a category of proof-relevant logical predicates or "'deliverables"' in the sense of (Burstall and McKinna 1993).

6.5 Interpretation of the syntax in a naive F-doctrine internal to $\mathit{Fam}(\mathbf{E})$

From Section 6.3 it is clear how to interpret the syntax in a naive F-doctrine internal to $\mathit{Fam}(\mathbf{E})$. For subsequent use it is, however, appropriate to spell out the particular nature of this interpretation and to introduce some notation. Let $\mathcal{C}$ be a naive F-doctrine in $\mathit{Fam}(\mathbf{E})$ (for $\mathbf{E}$ a model of the metalanguage). As $\mathit{Fst} : \mathit{Fam}(\mathbf{E}) \to \mathbf{E}$ preserves all the structure of a model of the metalanguage, in particular substitution and Π-types, the component-wise application of Fst to $\mathcal{C}$ furnishes a model $\mathit{Fst}(\mathcal{C})$ internal to $\mathbf{E}$. More precisely, we have $|\mathit{Fst}(\mathcal{C})| = \mathit{Fst}(|\mathcal{C}|)$ and so forth. From the definition of Fst and $[\![-]\!]$ it is immediate that the syntactic component of the interpretation in $\mathcal{C}$ coincides with the interpretation in $\mathit{Fst}(\mathcal{C})$. We abbreviate $([\![-]\!]^{\mathcal{C}})^{sem}$ by $[\![-]\!]^{\mathcal{C}}$.

Let $(\vec{X}, \vec{\mathbf{X}}) : (\mathit{Ty}^{\mathcal{C}})^n$, i.e. $\vec{X} : (\mathit{Ty}^{\mathit{Fst}\mathcal{C}})^n$ and $\vec{\mathbf{X}} : ((\mathit{Ty}^{\mathcal{C}})^{sem})^n(\vec{X})$. For $n \vdash \Gamma$ we have:

$$[\![n \vdash \Gamma]\!]^{\mathcal{C}}(\vec{X}, \vec{\mathbf{X}}) : |\mathcal{C}|^{sem}([\![n \vdash \Gamma]\!]^{\mathit{Fst}(\mathcal{C})}(\vec{X}))$$

Similarly for types. A term $\Gamma \vdash_n t : A$ gets interpreted as

$$[\![\Gamma \vdash_n t : A]\!]^{\mathcal{C}}(\vec{X}, \vec{\mathbf{X}}) = \mathcal{C}^{sem}([\![\Gamma \vdash t : A]\!]^{\mathit{Fst}(\mathcal{C})})$$

7 Sconing of a naive F-doctrine

Fix a naive F-doctrine $\mathcal{C}$ living in some model of type theory $\mathbf{E}$. Our aim is to define a new doctrine $\mathcal{S}c = \mathcal{S}c(\mathcal{C})$ internal to $\mathit{Fam} = \mathit{Fam}(\mathbf{E})$ in such a way that the model $\mathcal{C}$ arises as $\mathit{Fst}(\mathcal{S}c(\mathcal{C}))$.

Therefore, there is no need to define the syntactic components as they are fixed by this specification. Accordingly, we omit the ^{sem} superscript in the definition of the semantic components.

For $\Gamma\colon|\mathcal{C}|$ we write $\mathit{Sect}(\Gamma)$ for $\mathcal{C}(1,\Gamma)$—the set of global elements of Γ. We define

$$|\mathcal{S}c|(\Gamma\colon|\mathcal{C}|) = \mathit{Sect}(\Gamma) \to \mathit{Prop}$$

This means that the object of objects takes the form $(|\mathcal{C}|, \lambda\Gamma\colon|\mathcal{C}|.\mathit{Sect}(\Gamma) \to \mathit{Prop})$. The semantic component of the family of morphisms in $\mathcal{S}c$ is defined by

$$\mathcal{S}c(\Gamma\colon|\mathcal{C}|, \Gamma\colon\mathcal{S}c(\Gamma), \Delta\colon|\mathcal{C}|, \mathbf{\Delta}\colon\mathcal{S}c(\Delta), f\colon\mathcal{C}(\Gamma,\Delta)) =$$
$$\Pi\gamma\colon\mathit{Sect}(\Gamma).\Gamma(\gamma) \to \mathbf{\Delta}(f\circ\gamma)$$

The object of types $\mathit{Ty}^{\mathcal{S}c}$ is the restriction of $|\mathcal{S}c|$ to types, i.e. we have $\mathcal{S}c^{\mathit{Ty}}(A\colon\mathit{Ty}^{\mathcal{C}}) = \mathit{Sect}(A) \to \mathit{Prop}$.

We come to composition. If $\Gamma\colon\mathcal{S}c(\Gamma)$, $\mathbf{\Delta}\colon\mathcal{S}c(\Delta)$, $\mathbf{\Theta}\colon\mathcal{S}c(\Theta)$, and $f\colon\mathcal{C}(\Gamma,\Delta)$ and $g\colon\mathcal{C}(\Delta,\Theta)$ and furthermore $\mathbf{f}\colon\mathcal{S}c(\Gamma,\mathbf{\Delta},f)$ and $\mathbf{g}\colon\mathcal{S}c(\mathbf{\Delta},\mathbf{\Theta},g)$ then we define the semantic component of the composition of $\mathbf{f}$ and $\mathbf{g}$ by

$$\lambda\gamma\colon\mathit{Sect}(\Gamma).\lambda\boldsymbol{\gamma}\colon\Gamma(\gamma).\mathbf{g}(f\circ\gamma, \mathbf{f}(\gamma,\boldsymbol{\gamma})) : \mathbf{\Theta}((g\circ f)\circ\gamma)$$

Associativity of composition in $\mathcal{C}$ is required for this to "type-check". Similarly, we define identity morphisms and the axioms for a category are readily verified.

Notice that this construction amounts to the well-known sconing construction (glueing along the global sections functor) as discussed extensively in (Lambek and Scott 1985; Crole 1993). There the glued category lives inside **Set**, but as there is a structure-preserving functor from the glued category back to the original one this amounts to specifying a category internal to $\mathit{Fam}(\mathbf{Set})$. The construction of the products and exponentials in our $\mathcal{S}c$ is as in the abovementioned sources. We give their definitions on objects but omit the associated morphisms and the verifications.

For $\Gamma \in \mathcal{S}c(\Gamma)$ and $\mathbf{A} \in \mathit{Ty}^{\mathcal{S}c}(A)$ and $\mathbf{B} \in \mathit{Ty}^{\mathcal{S}c}(B)$ we define

$$(\Gamma.\mathbf{A})(\langle\gamma, x\rangle \in \mathit{Sect}(\Gamma.A)) = \Gamma(\gamma) \times \mathbf{A}(x)$$

$$(\mathbf{A} \Rightarrow \mathbf{B})(f\colon\mathit{Sect}(A \Rightarrow B)) = \Pi x\colon\mathit{Sect}(A).\mathbf{A}(x) \to \mathbf{B}(\mathrm{app}(f,x))$$

Now we will show that $\mathcal{S}c$ has the required Ty-indexed products as a category internal to Fam. Let $B : \mathit{Ty}^{\mathcal{C}} \to \mathit{Ty}^{\mathcal{C}}$ together with $\mathbf{B} : \Pi X\colon\mathit{Ty}^{\mathcal{C}}.\mathit{Ty}^{\mathcal{S}c}(X) \to \mathit{Ty}^{\mathcal{S}c}(B(X))$ be an endomorphism of the object of types in $\mathcal{S}c$. We claim that the semantic component of the product is given by

$$\forall(B, \mathbf{B}, t\colon\mathit{Sect}(\forall(B))) = \Pi X\colon\mathit{Ty}^{\mathcal{C}}.\Pi\mathbf{X}\colon\mathit{Ty}^{\mathcal{S}c}.\mathbf{B}(X, \mathbf{X}, \mathrm{App}(X)\circ t)$$

It remains to define type abstraction and application and to check the equations. For $X\colon Ty^{\mathcal{C}}$ and $\mathbf{X}\colon Ty^{\mathcal{S}c}(X)$ the semantic component of the application morphism is defined by

$$\mathbf{App}(X,\mathbf{X}) = \lambda t\colon Sect(\forall(B)).\lambda\mathbf{t}\colon \forall(B,\mathbf{B},t).\mathbf{t}(X,\mathbf{X}) : \mathbf{X}(\mathrm{App}(X)\circ t)$$

Conversely, if $\Gamma\colon \mathcal{S}c(\Gamma)$ and $f\colon \Pi X\colon Ty^{\mathcal{C}}.\mathcal{C}(\Gamma, B(X))$ and

$$\mathbf{f} : \Pi X\colon Ty^{\mathcal{C}}.\Pi\mathbf{X}\colon Ty^{\mathcal{S}c}(X).\mathcal{S}c(\Gamma,\Gamma,B(X),\mathbf{B}(X,\mathbf{X}),f(X))$$

is a family of morphisms in $\mathcal{S}c$ then we define

$$\begin{aligned}\mathbf{\Lambda}(\mathbf{f}) = \lambda\gamma\colon Sect(\Gamma).\lambda\boldsymbol{\gamma}\colon\Gamma(\gamma).\lambda X\colon Ty^{\mathcal{C}}.\lambda\mathbf{X}\colon Ty^{\mathcal{S}c}(X).\mathbf{f}(\gamma,\boldsymbol{\gamma},X,\mathbf{X}) :\\ \mathcal{S}c(\Gamma,\Gamma,\forall(B),\forall(B,\mathbf{B}),\Lambda(f))\end{aligned}$$

For this to "type-check" the identity $\mathrm{App}(X)\circ\Lambda(f)\circ\gamma = f(X)\circ\gamma$—a consequence of Ty-Beta has been used. The required equations are straightforward.

7.1 Interpretation of the syntax in $\mathcal{S}c(\mathcal{C})$

By construction of $\mathcal{S}c(\mathcal{C})$ one has $Fst(\mathcal{S}c(\mathcal{C})) = \mathcal{C}$. Thus, due to the analysis in Section 6.4, the interpretation in $\mathcal{S}c(\mathcal{C})$ gives rise to a (many-sorted) function $[\![-]\!]^{\mathcal{S}c}$ which maps

- A judgement $n \vdash \Gamma$ to a function

$$[\![n\vdash\Gamma]\!]^{\mathcal{S}c} : \Pi\vec{X}\colon Ty^n.\Pi\vec{\mathbf{X}}\colon (Ty^{\mathcal{S}c})^n(\vec{X}).Sect([\![n\vdash\Gamma]\!]^{\mathcal{C}}(\vec{X}))\to Prop$$

- A judgement $\Gamma\vdash_n t : A$ to a function

$$\begin{aligned}&[\![n\vdash t:A]\!]^{\mathcal{S}c} : \Pi\vec{X}\colon Ty^n.\Pi\vec{\mathbf{X}}\colon (Ty^{\mathcal{S}c})^n(\vec{X}).\\ &\Pi\gamma\colon Sect([\![n\vdash\Gamma]\!]^{\mathcal{C}}(\vec{X})).\Pi\boldsymbol{\gamma}\colon [\![n\vdash\Gamma]\!]^{\mathcal{S}c}(\vec{X},\vec{\mathbf{X}},\gamma).[\![n\vdash A]\!]^{\mathcal{S}c}(\vec{X},\vec{\mathbf{X}},[\![\Gamma\vdash_n t:A]\!]^{\mathcal{C}}\circ\gamma)\end{aligned}$$

Again, the equational theory is respected in the sense that convertible terms are mapped to the same semantic objects.

8 Exploring $\hat{\mathbb{V}}$

We will now explore the specific features of the presheaf category $\hat{\mathbb{V}}$. Our aim is to isolate a context of definitions and equations which enables to carry out the construction of the model $\mathcal{GT}$ entirely within the internal language of $\hat{\mathbb{V}}$, i.e. using the constructive metalanguage augmented by this context of definitions. We start by lifting the term model $\mathbb{T}$ to a naive F-doctrine $\mathcal{T}$ inside $\hat{\mathbb{V}}$ equipped with a higher-order abstraction operator. Secondly, we define sets and constructors of normal forms and neutral terms together with the associated embeddings.

8.1 The term model

Before giving the explicit definition of $\mathcal{T}$ we describe its construction in more abstract terms. This enables us to use existing results for the verification that $\mathcal{T}$ actually is a naive F-doctrine internal to $\hat{\mathbb{V}}$. Afterwards, we give the explicit definitions of (most of the) structure components of $\mathcal{T}$ for further reference and ease of the reader.

The abstract categorical construction of $\mathcal{T}$ proceeds in two steps. We start from the split fibred ConCCC $\mathbb{T} : \mathbb{S}^{\mathrm{op}} \to \mathbf{Cat}$ with $Cons_1$-products. Following (Asperti and Martini 1992) we call such a structure an *external model* for further reference. To this fibration we apply the *constant family* construction from (Jacobs 1991, §4.4.4) together with restriction along $\mathbb{V}$ resulting in an external model $\mathbb{T}//\mathbb{V}$ now over $\mathbb{V}$ and with type object $[1, \diamond] \in \mathbb{V}$. The objects of $(\mathbb{T}//\mathbb{V})_{[n,\Gamma]}$ are the objects of $\mathbb{T}_n$; a $(\mathbb{T}//\mathbb{V})_{[n,\Gamma]}$-morphism from Δ to Θ is a $\mathbb{T}_n$-morphism from $\Gamma.\Delta$ to Θ.

The definition of composition and identities in $\mathbb{T}//\mathbb{V}$ uses the cartesian products in $\mathbb{T}$. For example, the identity morphism at $\Delta \in (\mathbb{T}//\mathbb{V})_{[n,\Gamma]}$ is the product projection (in $\mathbb{T}_n$) from $\Gamma.\Delta$ to Δ. The composition of $f \in \mathbb{T}//\mathbb{V}_{[n,\Gamma]}(\Delta, \Theta)$ (hence $f \in \mathbb{T}_n(\Gamma.\Delta, \Theta)$) with $g \in \mathbb{T}//\mathbb{V}_{[n,\Gamma]}(\Theta, \Upsilon)$ is given by the morphism $g \circ \langle \pi_{\Gamma,\Delta}, f \rangle$ in $\mathbb{T}_n$.

Products and exponentials are inherited from $\mathbb{T}$.

From the development in loc. cit. adapted to the case of ConCCC's together with the fact that $\mathcal{I} : \mathbb{V} \to \mathbb{T}$ preserves finite products it follows that this is a split fibred ConCCC.

We now show directly that $\mathbb{T}//\mathbb{V}$ is an external model. The generic object is given by $0 \in \mathbb{V}_{[1,\diamond]}$. Recall from Section 2 that the product in $\mathbb{V}$ of $[n, \Gamma]$ and $[1, \diamond]$ is $[n+1, \Gamma^+]$. If $B \in \mathbb{T}//\mathbb{V}_{[n+1,\Gamma^+]}$ then we have $\forall(B) \in \mathbb{T}//\mathbb{V}_{[n,\Gamma]}$. Furthermore, if $\Delta \in \mathbb{T}//\mathbb{V}_{[n,\Gamma]}$ and $f : \mathbb{T}//\mathbb{V}_{[n+1,\Gamma^+]}(\Delta^+, B)$, i.e. $f : \mathbb{T}//\mathbb{V}_{n+1}(\Gamma^+.\Delta^+, B)$, then $\Lambda_{\Gamma.\Delta,B}(f) : \mathbb{T}//\mathbb{V}_{[n,\Gamma]}(\Delta, \forall(B))$ is the transpose of f for the desired adjunction. The counit of the adjunction is given by application, i.e. $\mathrm{App}(0_{\Gamma^+.\forall(B)^+}, 0) : \mathbb{T}//\mathbb{V}_{[n+1,\Gamma^+]}(\forall(B)^+, B)$.

The desired model $\mathcal{T}$ is now obtained by applying the process of *internalisation* from (Asperti and Martini 1992) to the external model $\mathbb{T}//V$ resulting in a naive F-doctrine internal to $\hat{\mathbb{V}}$. For further reference and for the reader not familiar with loc. cit. we explicitate parts of the raw data of $\mathcal{T}$.

Its object of contexts is the presheaf Con defined by

$$Con_{[n,\Gamma]} = \{\Delta \mid n \vdash \Delta\} (= |\mathbb{T}_n|)$$
$$Con_{[\vec{A},w]}(\Delta) = \Delta[\vec{A}] \qquad \text{if } [\vec{A}, w] \in \mathbb{V}([m, \mathrm{B}], [n, \Gamma]) \text{ and } \vdash_n \Delta$$

The sub-presheaf of types $Ty \subset Con$ is defined by

$$Ty_{[n,\Gamma]} = \{A \mid \vdash_n A\} (= Ty^{\mathbb{T}_n})$$

Its family of morphisms $Mor : Con \times Con \to Type$ is given by

$$Mor_{[n,\Gamma]}(\Delta, \Theta) = \mathbb{T}_n(\Gamma.\Delta, \Theta)$$

as convertibility classes of syntactic context morphisms. If $[\vec{A}, w] \in \mathbb{V}([m, \mathrm{B}], [n, \Gamma])$ and $\vec{t} \in Mor_{[n,\Gamma]}(\Delta, \Theta)$ then we define

$$Mor_{[\vec{A},w]}(\vec{t}) = \vec{t}[\vec{A}] \circ (w.\Delta[\vec{A}])$$

From (Asperti and Martini 1992) the following result is now immediate.

Theorem 8 *The category $\mathcal{T}$ is a naive F-doctrine internal to $\hat{\mathbb{V}}$.*

Remark 9 *The presheaf $Sect(\Delta) = Mor(\diamond, \Delta)$ has the following remarkable property. For $([n, \Gamma], \Delta) \in \hat{\mathbb{V}}(Con)$ we have*

$$Sect_{[n,\Gamma]}(\Delta) = \mathbb{T}_n(\Gamma, \Delta)$$

so that the global sections in $\mathcal{T}$ correspond to vectors of open terms.

In order to explicitate the higher-order operations corresponding to type quantification and abstraction we require the following analysis of exponentiation with representable presheaves also used in (Asperti and Martini 1992).

Lemma 10 *Let $\mathbf{K}$ be a category, A be an object of $\mathbf{K}$, such that the products $I \times A$ exist for all $I \in \mathbf{K}$ and $F \in \hat{\mathbf{K}}$ be a presheaf. The exponential $\mathbf{y}(A) \to F$ of F by the representable presheaf $\mathbf{y}(A) = \mathbf{K}(-, A)$ is isomorphic to the functor $F_{- \times A}$.*

Proof. Immediate from the definition of exponentials in presheaf categories and the fact that the Yoneda embedding preserves products. □

Lemma 11 *Let F be a presheaf in $\hat{\mathbb{V}}$. The exponential $Ty \to F$ in $\hat{\mathbb{V}}$ is isomorphic to the presheaf F^{Ty} given by $F^{Ty}_{[n,\Gamma]} = F_{[n+1,\Gamma^+]}$.*

Proof. The presheaf Ty is isomorphic to the representable presheaf $\mathbf{y}([1,\diamond])$. We conclude with Lemma 10 □

To keep the exposition simple we shall henceforth treat the isomorphisms guaranteed by the above results as identities.

We now construct the required function $\forall : (Ty \to Ty) \to Ty$ in a pointwise way. Assume $[n,\Gamma] \in \mathbb{V}$ and $B : (Ty \to Ty)_{[n,\Gamma]}$. From Lemma 11 we know that in fact $B : Ty_{[n+1,\Gamma^+]}$, i.e. $n+1 \vdash B$. We put

$$\forall_{[n,\Gamma]}(B) = \forall(B) : \{X \mid n \vdash X\} = Ty_{[n,\Gamma]}$$

Naturality is immediate from the compatibility of type quantification with substitution and weakening.

In a similar way we define type abstraction and application following the settings in the external version $\mathbb{T}//\mathbb{V}$.

Proposition 12 (Interpretation in $\mathcal{T}$) *Suppose that* $[m,\Delta] \in \mathbb{V}$ *and* $\vec{X} \in \mathbb{S}(m,n)(= Ty^n_{[m,\Delta]})$. *If* $n \vdash \Gamma$ *then*

$$[\![n \vdash \Gamma]\!]_{[m,\Delta]}(\vec{X}) = \Gamma[\vec{X}]$$

Furthermore, if $\Gamma \vdash_n t : A$ *then*

$$[\![\Gamma \vdash_n t : A]\!]^{\mathcal{T}}_{[m,\Delta]}(\vec{X}) = [t[\vec{X}] \circ \pi_{\Gamma[\vec{X}],\Delta}]_{conv} \in \mathbb{T}_m(\Gamma[\vec{X}].\Delta\,,\ A[\vec{X}])$$

In particular, if $\diamond \vdash_0 t : A$ *is a closed term then*

$$[\![\diamond \vdash_0 t : A]\!]^{\mathcal{T}}_{[0,\diamond]}(\star) = [t]_{conv}$$

Proof. Induction on derivations. □

8.2 The presheaf of variables

The term model as a naive F-doctrine can more simply be defined internal to $\hat{\mathbb{S}}$ by applying internalisation directly to the external model $\mathbb{T}$. But that would not allow to have *open* terms of type

A,i.e. *generalised* elements of A in $\mathbb{T}$, being represented by $Sect(A)$. It is this property that opens up the possibility to use the simple-minded sconing construction from Section 7 to prove existence of normal forms and not only of weak head normal forms as was done in (Coquand and Dybjer 1996). However, in order to make use of this fact without having to look at $\hat{\mathbb{V}}$ externally, we have to express this property in terms of the internal language. This will be done by reformulating λ-abstraction in $\mathcal{T}$ as a higher-order operation in $\hat{\mathbb{V}}$.

Had we worked internal to $\hat{\mathbb{T}}$ then we would have a canonical isomorphism between $Sect(A) \to Sect(B)$ and $Sect(A \Rightarrow B)$ by an analogue of Lemma 11 since $Sect(A)$ is representable, i.e. $\mathcal{T}$ would be a *full internal subcategory* of $\hat{\mathbb{T}}$. Unfortunately, $Sect(A)$ is not representable in $\hat{\mathbb{V}}$. Yet, there exists a representable sub-presheaf $Par(A)$ of $Sect(A)$ corresponding to those open terms which are variables. This presheaf has the property that there is an isomorphism $\ell : (Par(A) \to Sect(B)) \to Sect(A \Rightarrow B)$ which is not as comfortable as a full internal subcategory but will suffice for our purposes. The reason for accepting this inconvenience is that normal forms are not closed under arbitrary substitutions and thus do not give rise to presheaves in $\hat{\mathbb{T}}$ but only in $\hat{\mathbb{V}}$.

Definition 13 *The presheaf* $Par : \hat{\mathbb{V}}(Ty)$ *is defined on objects by*

$$Par_{[n,\Gamma]}(A) = \mathbb{V}_n(\Gamma, A)$$

and for $[\vec{A}, w] \in \mathbb{V}([n', \Gamma'], [n, \Gamma])$ *and* $n \vdash A$ *and* $w' \in \mathbb{V}_n(\Gamma, A)$ *we define*

$$Par_{[\vec{A},w]}(w') = w'[\vec{A}] \circ w \in Par_{[n',\Gamma']}(A[\vec{A}])$$

The natural transformation $par : Par \to Sect$ *in* $\hat{\mathbb{V}}(Ty)$ *is defined as the inclusion from weakenings into terms.*

Lemma 14 *The presheaf* Par *is represented by the object* $([1,0],0)$ *of* $El(Ty)$. *For any object* $([n,\Gamma],A)$ *of* $El(Ty)$ *the object* $([n,\Gamma.A],A)$ *is the cartesian product of* $([n,\Gamma],A)$ *and* $([1,0],0)$.

Proof. Suppose $([n,\Gamma],A)$ is an object of $El(Ty)$, i.e. $n \vdash \Gamma$ and $n \vdash A$. By definition, an element of $Par_{[n,\Gamma]}(A)$ is a renaming $w \in \mathbb{V}_n(\Gamma, A)$. It gives rise to the morphism $[A, w] \in \mathbb{V}([n,\Gamma], [1,0])$ because $0[A] = A$. Furthermore, by using $0[A] = A$ again, we have that $[A, w]$ is in fact an $El(Ty)$-morphism from $([n,\Gamma],A)$ to $([1,0],0)$. For the converse, assume that $[B, w]$ is an arbitrary $El(Ty)$-morphism from $([n,\Gamma],A])$ to $([1,0],0)$, i.e. $[B, w] \in \mathbb{V}([n,\Gamma], [1,0])$ and $0[B] = A$ then we must have $B = A$, thus $w \in \mathbb{V}_n(\Gamma, A) = Par_{[n,\Gamma]}(A)$. It is obvious that this establishes a natural isomorphism between Par and $\mathbf{y}(([1,0],0))$.

For the second part we notice that Ty is representable by $[1,\diamond]$ and therefore the category $El(Ty)$ is isomorphic to the slice category $\mathbb{V}/[1,\diamond]$. So the cartesian product in question amounts to the pullback of $[A,\langle\rangle]:[n,\Gamma]\to[1,\diamond]$ and $[id_1,\langle\rangle]:[1,0]\to[1,\diamond]$. This pullback can be obtained using a general construction, but we prefer to give it directly as follows:

$$\begin{array}{ccc} [n,\Gamma.A] & \xrightarrow{[A,\pi'_{\Gamma,A}]} & [1,0] \\ {\scriptstyle [id_n,\pi_{\Gamma,A}]}\downarrow & & \downarrow{\scriptstyle [id_1,\langle\rangle]} \\ [n,\Gamma] & \xrightarrow[[A,\langle\rangle]]{} & [1,\diamond] \end{array}$$

To see that the upper arrow is indeed a $\mathbb{V}$-morphism notice again that $0[A]=A$. It is clear that the square commutes. If $[\vec{A},w]:[m,\Delta]\to[n,\Gamma]$ and $[B,w']:[m,\Delta]\to[1,0]$ is a cone then we must have $w\in\mathbb{V}_m(\Delta,\Gamma[\vec{A}])$ and $w'\in\mathbb{V}_m(\Delta,B)$ and (from the commutativity) $B=A[\vec{A}]$. The uniqe mediating morphism is then given by $[\vec{A},\langle w,w'\rangle]:[m,\Delta]\to[n,\Gamma.A]$. □

Corollary 15 *For every presheaf $F\in\hat{\mathbb{V}}(Ty)$ the exponential $Par\to F$ is isomorphic to the presheaf which sends $([n,\Gamma],A)$ to $F_{[n,\Gamma.A]}(A)$.*

Proof. Immediate from Lemma 14 and Lemma 10. □

Proposition 16 *There exists a natural transformation $\ell:\Pi A,B{:}\,Ty.(Par(A)\to Sect(B))\to Sect(A\Rightarrow B)$ such that the following equation holds in $\hat{\mathbb{V}}$*

$$\forall A,B{:}\,Ty.\forall t{:}\,Sect(A\Rightarrow B)\;.\quad t=\ell(\lambda p{:}\,Par(A).\mathrm{app}(t,par(p)))$$

Proof. By Cor. 15 the arguments to ℓ at stage $[n,\Gamma]$ amount to types $n\vdash A,B$ and a term t with $\Gamma.A\vdash_n t:B$. We define $\ell_{[n,\Gamma]}(A,B,t)$ as $\lambda_{\Gamma,A,B}(t)$ and the required equation follows from ETA. □

As announced before ℓ is actually an isomorphism, i.e. for $f:Par(A)\to Sect(B)$ we have $f=\lambda p{:}\,Par(A).\mathrm{app}(\ell(f),p)$. However, we do not require this property for our purposes.

Remark 17 *The presheaf Par together with ℓ can serve as an interpretation of the higher-order syntax for λ-calculus as described in (Despeyroux, Felty, and Hirschowitz 1995). As stated before, in a term model internal to $\hat{\mathbb{T}}$ the presheaf $Sect(A) \to Sect(B)$ is isomorphic to $Sect(A \Rightarrow B)$ and one thus obtains a model for the usual higher-order abstract syntax of typed λ-calculus as described e.g. in (Harper, Honsell, and Plotkin 1993).*

8.3 The presheaves of normal and neutral terms

As neutral terms and normal forms are stable under type substitutions and weakenings these sets induce presheaves over *Con* (and also over *Ty*) defined by

$$NF_{[n,\Gamma]}(\Delta) = \text{Tuples of normal forms of type } \Delta \text{ in context } \Gamma$$
$$NE_{[n,\Gamma]}(\Delta) = \text{Tuples of neutral terms of type } \Delta \text{ in context } \Gamma$$

The clauses of the inductive definition of these sets induce natural transformations as given in the second part of Figure 8.3. These operations allow to build up normal forms and neutral terms in the internal language. Again, we have higher-order constructors giving normal forms of functional and polymorphic types. Notice that we do not have an induction principle for NE and NF in $\hat{\mathbb{V}}$. We also have functions $i : \Pi A\colon Ty.NF(A) \to Sect(A)$ and $k : \Pi A\colon Ty.NE(A) \to NF(A)$ and therefore also $j : \Pi A\colon Ty.NE(A) \to Sect(A)$ defined as the composition $i \circ k$. The functions j and i commute with the constructors in the obvious way. For instance, if $p : Par(A)$ then

$$j(par_{NE}(p)) = par(p)$$

and if $B\colon Ty \to Ty$ and $f\colon \Pi X\colon Ty.NF(B(X))$ then

$$i(\Lambda_{NF}(f)) = \Lambda(\lambda X\colon Ty.i(f(X)))$$

In the case of λ_{NF} the higher-order abstractor ℓ is used: If $A, B\colon Ty$ and $f\colon Par(A) \to NF(B)$ then

$$i(\lambda_{NF}(A, B, f)) = \ell(j(ap_{NE}(u, v)))$$

Analogous equations hold for the remaining constructors of NE and NF.

We stress that the coercion i is not monic. For example

$$M_1 := \Lambda_{NF}(\lambda X\colon Ty.\lambda_{NF}(\lambda p\colon Par(X \Rightarrow X).k(par_{NE}(p))))$$

and

$$M_2 := \Lambda_{NF}(\lambda X\colon Ty.\lambda_{NF}(\lambda p\colon Par(X \Rightarrow X).\lambda_{NF}(\lambda x\colon Par(X).k(ap_{NE}(par_{NE}(p), k(par_{NE}(x)))))))$$

Externally, we have $M_1 = \Lambda X.\lambda p.p$ and $M_2 = \Lambda X.\lambda p.\lambda x.\mathrm{app}(p,x)$. These two are different normal forms of type $\forall X.X \Rightarrow X$ but $i(M_1) = i(M_2)$ by ETA.

We do not need that i is monic for the purposes of our proof; the fact that there can be different normal forms of the same conversion class simply means that our normalisation function will not be surjective, but simply picks out one particular normal form (in our example this will be M_2.) One could enforce uniqueness of normal forms by using an analogue of Huet's *long $\beta\eta$-normal forms* (Huet 1976) which is, however, not unproblematic since these are not stable under type substitution. So the morphism part of the presheaf NF would have to perform certain η-expansions.

This completes the inspection of the internal language of $\hat{\mathbb{V}}$. The construction of the glued model to follow can now be carried out entirely on top of the definitions made so far and summarised in Figure 8.3.

9 The Tait model

Our aim is now to apply a mild variant of the sconing construction from Section 7 to the term model $\mathcal{T}$. We do not consider arbitrary predicates over sections, but only those which are valid for all sections corresponding to neutral terms and which imply the existence of normal forms, i.e. a naive version of Tait's computability predicates. More precisely, we define a naive F-doctrine $\mathcal{GT}$ internal to $\mathit{Fam}(\hat{\mathbb{V}})$ the first component of which is the term model $\mathcal{T}$, i.e. $\mathit{Fst}(\mathcal{GT}) = \mathcal{T}$. The second component of the object of contexts $\mathit{Con}^{\mathcal{GT}} : \mathit{Con} \to \mathit{Type}$ is defined as

$$\begin{aligned}&\lambda\Gamma\colon \mathit{Con}.\Sigma P\colon \mathit{Sect}(\Gamma) \to \mathit{Prop}.\\ &\quad(\Pi t\colon \mathit{Sect}(\Gamma).P(t) \to \{t'\colon \mathit{NF}(\Gamma) \mid i(t') = t\})\times\\ &\quad(\Pi u\colon \mathit{NE}(\Gamma).P(j(u)))\end{aligned}$$

If $\Gamma\colon \mathit{Con}^{\mathcal{GT}}(\Gamma)$ we refer to its three components by Γ^{pred}, $\mathtt{q}^\Gamma$, and $\mathtt{u}^\Gamma$, respectively. I.e. we have

$$\Gamma^{\mathit{pred}} : \mathit{Sect}(\Gamma) \to \mathit{Prop}$$

and

$$\mathtt{q}^\Gamma : \Pi t\colon \mathit{Sect}(\Gamma).\Gamma^{\mathit{pred}}(t) \to \mathit{NF}(\Gamma)$$

in such a way that $i(\mathtt{q}^\Gamma(t,\mathtt{t})) = t$. Finally, $\mathtt{u}^\Gamma : \Pi u\colon \mathit{NE}(\Gamma).\Gamma^{\mathit{pred}}(j(u))$.

According to our convention on arguments which can be inferred we often omit the first argument to $\mathtt{q}$.

$Con : Type$
$Ty : Type$
$Mor : Con \times Con \to Type$
$id : \Pi\Gamma\colon Con.Mor(\Gamma, \Gamma)$
$comp : \Pi\Gamma, \Delta, \Theta\colon Con.Mor(\Delta, \Theta) \times Mor(\Gamma, \Delta) \to Mor(\Gamma, \Theta)$
$\diamond : Con$
$\langle\rangle : \Pi\Gamma\colon Con.Mor(\Gamma, \diamond)$
$-.- : Con \times Ty \to Con$
$cons : \Pi\Gamma\colon Con.\Pi A\colon Ty.\Pi\Delta\colon Con.Mor(\Delta, \Gamma) \times Mor(\Delta, A) \to Mor(\Delta, \Gamma.A)$
$\pi : \Pi\Gamma\colon Con.\Pi A\colon Ty.Mor(\Gamma.A, \Gamma)$
$\pi' : \Pi\Gamma\colon Con.\Pi A\colon Ty.Mor(\Gamma.A, A)$
$\Rightarrow\colon Ty \times Ty \to Ty$
$\mathrm{app} : \Pi A, B\colon Ty.\Pi\Delta\colon Con.Mor(\Delta, A \Rightarrow B) \times Mor(\Delta, A) \to Mor(\Delta, B)$
$\lambda : \Pi A, B\colon Ty.\Pi\Delta\colon Con.Mor(\Delta.A, B) \to Mor(\Delta, A \Rightarrow B)$
$\forall : (Ty \to Ty) \to Ty$
$\mathrm{App} : \Pi B\colon Ty \to Ty.\Pi A\colon Ty.Mor(\forall(B), B(A))$
$\Lambda : \Pi B\colon Ty \to Ty.\Pi\Delta\colon Con.(\Pi X\colon Ty.Mor(\Delta, B(X))) \to Mor(\Delta, \forall(B))$

Equations stating that the above data define a naive F-doctrine

$Par : Ty \to Type$
$par : \Pi A\colon Ty.Par(A) \to Sect(A)$
$\ell : \Pi A, B\colon Ty.(Par(A) \to Sect(B)) \to Sect(A \Rightarrow B)$
$\forall A, B\colon Ty.\forall t\colon Sect(A \Rightarrow B).\quad t = \ell(\lambda p\colon Par(A).\mathrm{app}(t, par(p)))$

$NE, NF : Ty \to Type$
$par_{NE} : \Pi A\colon Ty.Par(A) \to NE(A)$
$ap_{NE} : \Pi A, B\colon Ty.NE(A \Rightarrow B) \to NF(A) \to NE(B)$
$Ap_{NE} : \Pi B\colon Ty \to Ty.NE(\forall(B)) \to \Pi X\colon Ty.NE(B(X))$
$proj_{NE} : \Pi\Gamma\colon Con.\Pi A\colon Ty.NE(\Gamma.A) \to NE(\Gamma)$
$var_{NE} : \Pi\Gamma\colon Con.\Pi A\colon Ty.NE(\Gamma.A) \to NE(A)$
$\langle\rangle_{NF} : NF(\diamond)$
$cons_{NF} : \Pi\Gamma\colon Con.\Pi A\colon Ty.NF(\Gamma) \to NF(A) \to NF(\Gamma.A)$
$\lambda_{NF} : \Pi A, B\colon Ty.(Par(A) \to NF(B)) \to NF(A \Rightarrow B)$
$\Lambda_{NF} : \Pi B\colon Ty \to Ty.(\Pi X\colon Ty.NF(B\ X)) \to NF(\forall(B))$
$i : \Pi A\colon Ty.NF(A) \to Sect(A)$
$j : \Pi A\colon Ty.NE(A) \to Sect(A)$
$k : \Pi A\colon Ty.NE(A) \to NF(A)$

Equations specifying the interaction of i and j with the constructors of NE and NF

Figure 2: Internal language of $\hat{\mathbb{V}}$

We define $Ty^{\mathcal{GT}} : Ty \to Type$ as the restriction of $Con^{\mathcal{GT}}$ to Ty.

The second component of the object of morphisms is defined by

$$Mor^{\mathcal{GT}}(\Gamma, \Gamma, \Delta, \mathbf{\Delta}, f\colon Mor(\Gamma, \Delta)) = \Pi\gamma\colon Sect(\Gamma).\Gamma^{pred}(\gamma) \to \mathbf{\Delta}^{pred}(f \circ \gamma)$$

Since the definition of $Mor^{\mathcal{GT}}$ does not depend on the added components $\mathtt{q}$ and $\mathtt{u}$, the construction of products, exponentials, and type-indexed products can be carried out as in the case of $\mathcal{Sc}(\mathcal{T})$ replacing $Ty^{\mathcal{Sc}}$ by $Ty^{\mathcal{GT}}$ in the case of the type-indexed products. It only remains to show that these constructions can be extended to "Tait-predicates", i.e. the predicates are defined as before, but we need to give the $\mathtt{q}$ and $\mathtt{u}$ components. For the ease of the reader we reproduce the predicates themselves, as well.

First, we define

$\diamond^{pred}(x\colon Sect(\diamond))$	$=$	$\{\star\}$
$\mathtt{q}^{\diamond}(x\colon \diamond^{pred}(x))$	$=$	$\langle\rangle_{NF}$
$\mathtt{u}^{\diamond}(u\colon NE(\diamond))$	$=$	$\star$

For $\Gamma\colon Con$, $\Gamma\colon Con^{\mathcal{GT}}(\Gamma)$, and $A\colon Ty$, and $\mathbf{A}\colon Ty^{\mathcal{GT}}(A)$ we define

$(\Gamma.\mathbf{A})^{pred}(\langle\gamma, x\rangle\colon Sect(\Gamma.A))$	$=$	$\Gamma^{pred}(\gamma) \times \mathbf{A}^{pred}(x)$
$\mathtt{q}^{\Gamma.\mathbf{A}}(\langle\boldsymbol{\gamma}, \mathbf{x}\rangle\colon (\Gamma.\mathbf{A})(\langle\gamma, x\rangle))$	$=$	$cons_{NF}(\mathtt{q}^{\Gamma}(\boldsymbol{\gamma}), \mathtt{q}^{\mathbf{A}}(\mathbf{x}))$
$\mathtt{u}^{\Gamma.\mathbf{A}}(u\colon NE(\Gamma.A))$	$=$	$\langle \mathtt{u}^{\Gamma}(proj_{NE}(u)), \mathtt{u}^{\mathbf{A}}(var_{NE}(u))\rangle$

We come to exponentials. Let $A, B\colon Ty$, and $\mathbf{A}\colon Ty^{\mathcal{GT}}(A)$ and $\mathbf{B}\colon Ty^{\mathcal{GT}}(B)$. We define

$(\mathbf{A} \Rightarrow \mathbf{B})^{pred}(t\colon Sect(A \Rightarrow B))$	$=$	$\Pi a\colon Sect(A).\mathbf{A}^{pred}(a) \to \mathbf{B}^{pred}(\mathrm{app}(t, a))$
$\mathtt{q}^{\mathbf{A}\Rightarrow\mathbf{B}}(\mathbf{t}\colon (\mathbf{A} \Rightarrow \mathbf{B})^{pred}(t))$	$=$	$\lambda_{NF}(\lambda p\colon Par(A).\mathtt{q}^{\mathbf{B}}(\mathbf{t}(p, \mathtt{u}^{\mathbf{A}}(par_{NE}(p)))))$
$\mathtt{u}^{\mathbf{A}\Rightarrow\mathbf{B}}(u\colon NE(A \Rightarrow B))$	$=$	$\lambda a\colon Sect(A).\lambda\mathbf{a}\colon \mathbf{A}^{pred}(a).\mathtt{u}^{\mathbf{B}}(ap_{NE}(u, \mathtt{q}^{\mathbf{A}}(\mathbf{a})))$

To type-check the *quote* expression, we need to show that

$$i(\lambda_{NF}(\lambda p\colon Par(A).\mathtt{q}^{\mathbf{B}}(\mathbf{t}(p, \mathtt{u}^{\mathbf{A}}(par_{NE}(p)))))) = t$$

We calculate as follows. (The first argument to $\mathtt{q}$ will be given.)

$$\begin{aligned}
& i(\lambda_{NF}(\lambda p\colon Par(A).\mathtt{q}^{\mathbf{B}}(\mathrm{app}(t, par(p))\,,\ \mathbf{t}(p, \mathtt{u}^{\mathbf{A}}(par_{NE}(p)))))) \\
= \quad & \ell(\lambda p\colon Par(A).i(\mathtt{q}^{\mathbf{B}}(\mathrm{app}(t, par(p))\,,\ \mathbf{t}(p, \mathtt{u}^{\mathbf{A}}(par_{NE}(p)))))) \\
= \quad & \ell(\lambda p\colon Par(A).\mathrm{app}(t, par(p))) \qquad \text{by equation for } \ell \\
= \quad & t
\end{aligned}$$

Finally the type-indexed products. We require a generic Tait predicate for every type X; more precisely we define $\mathtt{U} : \Pi X\colon Ty.\, Ty^{\mathcal{GT}}(X)$ by

$$\begin{array}{lcl} \mathtt{U}(X)^{pred}(t\colon Sect(X)) & = & \{t'\colon NF(X) \mid i(t') = t\} \\ \mathtt{q}^{\mathtt{U}(X)}(\mathbf{t}\colon \mathtt{U}(X)^{pred}(t)) & = & \mathbf{t} \\ \mathtt{u}^{\mathtt{U}(X)}(u\colon NE(X)) & = & k(u) \end{array}$$

Now for $B\colon Ty \to Ty$ and $bB\colon \Pi X\colon Ty.\, Ty^{\mathcal{GT}}(X) \to Ty^{\mathcal{GT}}(B(X))$ we define

$$\begin{array}{lcl} \forall(\mathbf{B})^{pred}(t\colon Sect(\forall(B))) & = & \Pi X\colon Ty.\Pi \mathbf{X}\colon Ty^{\mathcal{GT}}(X).\mathbf{B}(X,\mathbf{X})^{pred}(\mathrm{App}(t,X)) \\ \mathtt{q}^{\forall(\mathbf{B})}(\mathbf{t}\colon \forall(\mathbf{B})^{pred}(t)) & = & \Lambda_{NF}(\lambda X\colon Ty.\mathtt{q}^{\mathbf{B}(X,\mathtt{U}(X))}(\mathbf{t}(X,\mathtt{U}(X)))) \\ \mathtt{u}^{\forall(\mathbf{B})}(u\colon NE(\forall(B))) & = & \lambda X\colon Ty.\lambda \mathbf{X}\colon Ty^{\mathcal{GT}}(X).\mathtt{u}^{\mathbf{B}(X,\mathbf{X})}(Ap_{NE}(u,X)) \end{array}$$

10 The normalisation function

Theorem 18 *For every closed type $0 \vdash A$ there exists a constructively definable function nf_A mapping closed terms of type A to normal forms in such a way that if $0 \vdash t : A$ then $0 \vdash nf_A(t) = t : A$ and if $0 \vdash t = t' : A$ then $nf_A(t)$ and $nf_A(t')$ are syntactically equal.*

Proof. Consider the interpretation of the syntax in the model $\mathcal{GT}$. We make the following definitions (in the constructive metalanguage, no longer in $\hat{\mathbb{V}}$). Recall that $\mathbb{T}_0(\bullet, A)$ is the set $\{t \mid \bullet \vdash_0 t : A\}$ modulo convertibility.

$$\mathbf{A}^{pred}(t \in \mathbb{T}_0(\bullet, A)) = ([\![0 \vdash A]\!]^{\mathcal{GT}})^{pred}{}_{[0,\bullet]}(t, id_{[0,\bullet]}) \in Prop$$

$$\mathbf{A}^{quote}(t \in \mathbb{T}_0(\bullet, A)) = ([\![0 \vdash A]\!]^{\mathcal{GT}})^{quote}{}_{[0,\bullet]}(t) : \mathbf{A}^{pred}(t) \to \{t'\colon NF(A) \mid i(t') = t\}$$

Now, if $\vdash_0 t : A$ then $[\![\vdash_0 t : A]\!]^{\mathcal{GT}}_{[0,\bullet]} : \mathbf{A}^{pred}(t)$ and we define

$$nf_A(t) = \mathbf{A}^{quote}([\![\vdash_0 t : A]\!]^{\mathcal{GT}}_{[0,\bullet]})$$

These definitions are type-correct by Prop. 12.

From the type of $\mathbf{A}^{quote}$ we know that

$$i(nf_A(t)) = t$$

Furthermore, by the soundness of the interpretation in the Tait model the function nf maps convertible terms to equal normal forms. □

The restriction to closed types and terms is merely for ease of presentation. Open terms can be normalised by normalising their universal closures.

11 Conclusion

We have defined an internal model of system F the interpretation in which yields a normalisation function. The model uses the idea of glueing along the global sections functor ("Freyd sconing"), but had to be constructed internal to a category of presheaves where internal global sections correspond to open terms. In this way we were able to overcome the difficulties with full $\beta\eta$-reduction experienced in (Coquand and Dybjer 1996) where an ordinary set-theoretic semantic was used and thus reduction underneath an abstraction could not be catered for. Our method of using presheaves seems to be a general tool applicable in such situations. Furthermore, we have generalised the apporach in loc. cit. to polymorphism by considering presheaves not w.r.t. classical set theory, but relative to a model of impredicative type theory where polymorphic quantification has a natural interpretation as dependent product. We have employed the fact that the operation of passing to a presheaf model preserves the possibility of impredicative quantification cf. Prop. 6. Compared to (Altenkirch, Hofmann, and Streicher 1995) where presheaves were also used, we have in this paper used the internal language of the presheaf category for expressing the glueing construction. It appeared in the course of this work that due to the extra complication arising from type variables and polymorphism the argument is no longer manageable without using the internal language. The important point is that functional abstraction and type quantification are expressible as higher-order operations in this internal language. We have explained how this relates to *higher-order abstract syntax* for λ-calculus (Harper, Honsell, and Plotkin 1993; Despeyroux, Felty, and Hirschowitz 1995).

Our proof can be entirely formalised in extensional type theory, for instance Nuprl (Constable et al. 1986). One can also aim at a formalisation in intensional type theory like ALF or Lego, but then in order to define extensional objects like function spaces or the presheaf of convertibility classes one would have to use *setoids*, i.e. types with equivalence relations, as has been done in (Coquand 1994). Notice that such a formalisation would have to implement presheaves and natural transformations and not merely the internal language. All the definitions which for ease of exposition we have made in terms of the internal language have to be expanded into their set-theoretic or rather type-theoretic formulations.

From such a formalisation one can immediately derive an executable function *nf*. This function

nf is given by type-theoretic code and executed by the normaliser of the implemented type theory. It may seem that nothing has been gained because one normalisation procedure has been implemented in terms of another one. Note, however, that the normal forms from Section 2.3 are a simple inductive definition so that in order to compute them one only needs iterative weak head reduction for closed terms on the meta-level.

A more efficient program can be obtained by translating the type-theoretic code into a functional program, for instance in Standard ML. To do this one has to systematically replace dependently-typed inductive definitions by recursive datatypes and delete computationally-irrelevant arguments. For example the type of the semantic values which in the type-theoretic formulation depends on types and on (convertibility classes of) terms becomes in Standard ML the following recursive datatype where `Tm` is a simple inductive definition of terms.

```
datatype Sem = s_arr of int->Sem -> Sem |
               s_pi of int->((Sem -> Tm)*(Tm -> Sem)) -> Sem |
               s_u of Tm;
```

The integers are the "residuum" of the variable renamings in the type-theoretic formulation.

In order to justify these changes of data structure a general theory is called for. Lacking such theory at the moment we have to content ourselves with intuition and—if desired—a brute-force verification of the SML program using logical relations defining in which sense the SML program simulates the type-theoretic one. Having obtained such an SML program for normalisation of type-theoretic expression one has achieved the goal of employing the interpreter of the functional language for the purpose of normalisation although the interpreter itself performs weak-head reduction. Such an SML program exists and has been successfully tested on examples though has not been verified formally.

There are various ways in which the present work can be extended. A relatively easy task is the addition of inductive types such as a natural numbers type $\mathbf{N}$ with primitive recursion to $F_{\beta\eta}$. One has to adjust the definitions of normal forms and neutral terms and defines $\mathbf{nat}^{pred}$ as the property of being convertible to a normal form. The interpretation of zero and successor is then straightforward, for the interpretation of the recursor one uses primitve recursion over normal forms of $\mathbf{N}$ in $\hat{\mathbb{V}}$. Similarly, one can treat binary sum types albeit not with a general η-rule like in (Ghani 1995). Incorporating η-equality for sums constitutes a difficult problem and requires further research, yet we consider it an interesting task because usual term rewriting is not applicable to η-

equality for sum types as it would violate Church-Rosser, a property irrelevant for the reduction-free approach.

The other direction consists of extending the present method to dependently typed systems. Although there are no principle obstacles this requires considerable effort because the notions of model involved are more complex. In view of the results in the present paper adding impredicativity would be a straightforward matter once type dependency has been settled. For a restricted notion of equality (namely equality of whnfs) this programme has already been carried out by Per Martin-Löf (1975). Using the ideas of higher-order abstract syntax and presheaf models we hope to be able to extend Martin-Löf's result to full $\beta\eta$-equality.

References

Altenkirch, T., M. Hofmann, and T. Streicher (1995). Categorical reconstruction of a reduction-free normalisation proof. In D. Pitt and D. Rydeheard (Eds.), *Proc. CTCS '95, Springer LNCS, Vol. 953*, pp. 182–199.

Altenkirch, T., M. Hofmann, and T. Streicher (1996). Reduction-free normalisation for a polymorphic system. In *Proc. LICS '96, New Brunswick, N. J.* IEEE.

Asperti, A. and S. Martini (1992). Categorical models of polymorphism. *Information and Computation 99*, 1–79.

Berger, U. and H. Schwichtenberg (1991). An inverse of the evaluation functional for typed λ-calculus. In *Proceedings of LICS '91, Amsterdam*, pp. 203–211.

Burstall, R. and J. McKinna (1993). Deliverables: An approach to program semantics in constructions. In *Proc. MFCS '93, Springer LNCS, Vol. 711.* Also as LFCS technical report ECS-LFCS-91-133.

Constable, R. et al. (1986). *Implementing Mathematics with the Nuprl Development System.* Prentice-Hall.

Coquand, C. (1994). From semantics to rules: a machine-assisted analysis. In Börger, Gurevich, and Meinke (Eds.), *Proceedings of CSL '93*, pp. 171–185. Springer, LNCS.

Coquand, T. and P. Dybjer (1996). Intuitionistic model constructions and normalization proofs. *Mathematical Structures in Computer Science.* To appear. Previous version has appeared in

the informal proceedings of the BRA-Types workshop, 1993 held in Nijmegen.

Crole, R. (1993). *Categories for Types*. Cambridge University Press.

Despeyroux, J., A. Felty, and A. Hirschowitz (1995). Higher-order abstract syntax in Coq. In M. Dezani and G. Plotkin (Eds.), *Typed Lambda Calculi and Applications*, pp. 124–138. Springer LNCS vol. 902.

Ghani, N. (1995). $\beta\eta$-equality for coproducts. In *Proceedings of TLCA '95, Edinburgh*. Springer. LNCS 902.

Harper, R., F. Honsell, and G. Plotkin (1993, January). A framework for defining logics. *Journal of the ACM 40*(1), 143–184.

Huet, G. (1976). *Résolution d'équations dans des langages d'ordre* $1, 2, \ldots, \omega$. Ph. D. thesis, Université de Paris VII.

Jacobs, B. (1991). *Categorical Type Theory*. Ph. D. thesis, University of Nijmegen.

Lambek, J. and P. Scott (1985). *Introduction to Higher-Order Categorical Logic*. Cambridge University Press.

Luo, Z. (1994). *Computation and Reasoning*. Oxford University Press.

Luo, Z. and R. Pollack (1992). LEGO Proof Development System: User's Manual. Technical Report ECS-LFCS-92-211, University of Edinburgh.

Martin-Löf, P. (1975). An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Sheperdson (Eds.), *Logic Colloquium 1973*, pp. 73–118. North-Holland.

Moerdijk, I. and S. M. Lane (1992). *Sheaves in Geometry and Logic. A First Introduction to Topos Theory*. Springer.

Phoa, W. (1992). An introduction to fibrations, topos theory, the effective topos, and modest sets. Technical Report ECS-LFCS-92-208, LFCS Edinburgh.

Streicher, T. (1991). *Semantics of Type Theory*. Birkhäuser.