# Normalisation by Evaluation for Dependent Types

**Thorsten Altenkirch[1] and Ambrus Kaposi[1]**

1 **School for Computer Science, University of Nottingham**
**Nottingham, United Kingdom**
`{txa|auk}@cs.nott.ac.uk`

## Abstract

We develop normalisation by evaluation (NBE) for dependent types based on presheaf categories. Our construction is formulated using internal type theory using quotient inductive types. We use a typed presentation hence there are no preterms or realizers in our construction. NBE for simple types is using a logical relation between the syntax and the presheaf interpretation. In our construction, we merge the presheaf interpretation and the logical relation into a proof-relevant logical predicate. We have formalized parts of the construction in Agda.

**1998 ACM Subject Classification** F.4.1 Mathematical Logic

**Keywords and phrases** normalisation by evaluation, dependent types, internal type theory, logical relations, Agda

**Digital Object Identifier** 10.4230/LIPIcs.xxx.yyy.p

## 1 Introduction

### 1.1 Specifying normalisation

Normalisation can be given the following specification.

We denote the type of well typed terms of type $A$ in context $\Gamma$ by $\mathsf{Tm}\,\Gamma\,A$. This type is defined as a quotient inductive inductive type (QIIT, see [10]): in addition to normal constructors for terms such as $\mathsf{lam}$ and $\mathsf{app}$, it also has equality constructors eg. expressing the $\beta$ computation rule for functions. An equality $t \equiv_{\mathsf{Tm}\,\Gamma\,A} t'$ expresses that $t$ and $t'$ are convertible.

The type of normal forms is denoted $\mathsf{Nf}\,\Gamma\,A$ and there is an embedding from it to terms $\ulcorner - \urcorner : \mathsf{Nf}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A$. Normal forms are defined as a usual inductive type, decidability of equality is straightforward.

Normalisation is given by a function $\mathsf{norm}_A$ which takes a term to a normal form. It needs to be an isomorphism:

$$\text{completeness} \circlearrowleft \qquad \mathsf{norm}_A \downarrow \frac{\mathsf{Tm}\,\Gamma\,A}{\mathsf{Nf}\,\Gamma\,A} \uparrow \ulcorner - \urcorner \qquad \curvearrowright \text{stability}$$

If we normalise a term, we obtain a term which is convertible to it: $t \equiv \ulcorner \mathsf{norm}\, t \urcorner$. This is called completeness. The other direction is called stability: $n \equiv \mathsf{norm}\,\ulcorner n \urcorner$. It expresses that there is no redundancy in the type of normal forms. This property makes it possible to establish properties of the syntax by induction on normal forms.

Soundness, that is, if $t \equiv t'$ then $\mathsf{norm}\, t \equiv \mathsf{norm}\, t'$ is given by congruence of equality. The elimination rule for the QIIT of the syntax ensures that everything defined on the syntax respects soundness.


Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–16
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1.2 NBE for simple type theory

Normalisation by Evaluation (NBE) [6] is one way to implement this specification. It works by evaluating the syntax in a presheaf model over the category of renamings $\mathsf{REN}^{\mathsf{op}}$ and with normal forms for the interpretation of the base types. Note that for any context $\Gamma$ one can define the presheaves of terms, neutral terms (the subset of normal forms where an eliminator is applied to a variable) and normal forms. The action on objects is just returning substitutions, lists of neutral terms and lists of normal forms, respectively.

$$
\begin{aligned}
&\mathsf{Tm}_\Delta, \mathsf{Ne}_\Delta, \mathsf{Nf}_\Delta : \mathsf{PSh}\,(\mathsf{REN}^{\mathsf{op}})\\
&\mathsf{TM}_\Delta\,\Gamma := \mathsf{Tms}\,\Gamma\,\Delta\\
&\mathsf{NE}_\Delta\,\Gamma := \mathsf{Nes}\,\Gamma\,\Delta\\
&\mathsf{NF}_\Delta\,\Gamma := \mathsf{Nfs}\,\Gamma\,\Delta
\end{aligned}
$$

To normalise substitutions, one defines two natural transformations unquote $\mathsf{u}_\Delta$ and quote $\mathsf{q}_\Delta$ by induction on the structure of contexts and types such that the diagram in figure 1 commutes. $[\![\Delta]\!]$ denotes the interpretation of $\Delta$ in the presheaf model and $\mathsf{R}_\Delta$ denotes the logical relation at context $\Delta$ between $\mathsf{TM}_\Delta$ and $[\![\Delta]\!]$. The logical relation is equality at the base type.

$$
\begin{tikzcd}
\mathsf{NE}_\Delta \arrow[r, "\mathsf{u}_\Delta"] \arrow[dr, "\ulcorner\_\urcorner"'] & \Sigma\,(\mathsf{TM}_\Delta \times [\![\Delta]\!])\,\mathsf{R}_\Delta \arrow[r, "\mathsf{q}_\Delta"] \arrow[d, "\mathsf{proj}"] & \mathsf{NF}_\Delta \arrow[dl, "\ulcorner\_\urcorner"] \\
& \mathsf{TM}_\Delta &
\end{tikzcd}
$$

**Figure 1** The type of quote and unquote for a context $\Delta$ in NBE for simple types.

Now a subsitution $\sigma$ can be normalised by quote: it needs the substitution itself, the interpretation $[\![\sigma]\!]$ and a proof that they are related. This is given by the fundamental theorem of the logical relation denoted by $\mathsf{R}_\sigma$ which also needs two related elements: these are given by unquoting the identity substitution which is a neutral substitution.

$$\mathsf{norm}_\Delta\,(\sigma : \mathsf{TM}_\Delta\,\Gamma) : \mathsf{NF}_\Delta\,\Gamma := \mathsf{q}_{\Delta\,\Gamma}\,(\sigma, [\![\sigma]\!], \mathsf{R}_\sigma\,(\mathsf{u}_{\Gamma\,\Gamma}\,\mathsf{id}_\Gamma))$$

Completeness is given by commutativity of the right hand triangle. Stability can be proven by mutual induction on terms and normal forms.

A nice property of this approach is that the part of unquote and quote which gives $[\![\Delta]\!]$ can be defined separately from the part which gives relatedness, hence the normalisation function can be defined independently from the proof that it is complete.

## 1.3 NBE for type theory

In the case of simple type theory, types are closed, so they act like contexts. Quote at a type $A$ is just a natural transformation.

$$\mathsf{q}_A : \Sigma\,(\mathsf{TM}_A \times [\![A]\!])\,\mathsf{R}_A \mathrel{\dot{\to}} \mathsf{NF}_A.$$

In the case of type theory, types depend on contexts, so $\mathsf{TM}_{\Gamma\vdash A}$ becomes a family of presheaves over $\mathsf{TM}_\Gamma$, $[\![\Gamma \vdash A]\!]$ is a family over $[\![\Gamma]\!]$ and $\mathsf{R}_{\Gamma\vdash A}$ depends on $\mathsf{R}_\Gamma$ (and a term of that type and the interpretation of a term of that type).

$$\mathsf{TM}_{\Gamma\vdash A}, \mathsf{NE}_{\Gamma\vdash A}, \mathsf{NF}_{\Gamma\vdash A} : \mathsf{FamPSh}\,\mathsf{TM}_\Gamma$$
$$[\![\Gamma \vdash A]\!] : \mathsf{FamPSh}\,[\![\Gamma]\!]$$
$$\mathsf{R}_{\Gamma\vdash A} : \mathsf{FamPSh}\left(\Sigma\left(\Sigma\left(\mathsf{TM}_\Gamma \times [\![\Gamma]\!]\right)\mathsf{R}_\Gamma\right)\left(\mathsf{TM}_{\Gamma\vdash A} \times [\![\Gamma \vdash A]\!]\right)\right)$$

We can try to define quote and unquote for this type as a family of natural transformations. The type of quote and unquote omitting the naturality condition would be the following. These types encode the commutativity of the triangles as well.

$$\mathsf{q}_{(\Gamma\vdash A)\,\Psi} : (p : \mathsf{R}_{\Gamma\,\Psi}\,\rho\,\alpha)(t : \mathsf{TM}_A\,\rho)(v : [\![A]\!]\,\alpha) \to \mathsf{R}_A\,p\,t\,v \to \Sigma(n : \mathsf{NF}_A\,\rho).t \equiv \ulcorner n \urcorner$$
$$\mathsf{u}_{(\Gamma\vdash A)\,\Psi} : (p : \mathsf{R}_{\Gamma\,\Psi}\,\rho\,\alpha)(n : \mathsf{NE}_A\,\rho) \to \Sigma(v : [\![A]\!]\,\alpha).\mathsf{R}_A\,p\,\ulcorner n \urcorner\,v$$

However there seems to be no way to define quote and unquote this way because quote does not preserve the logical relation. The problem is that when defining unquote at $\Pi$ we need to define a semantic function which works for arbitrary inputs, not only those which are related to a term. It seems that we need to restrict the presheaf model to only contain such functions.

We solve this problem by replacing the presheaf and the logical relation by a proof relevant logical predicate. We denote the logical predicate at a context $\Delta$ by $[\![\Delta]\!]$. We define normalisation following the diagram in figure 2.

$$\begin{tikzcd}
\mathsf{NE}_\Delta \arrow[r, "\mathsf{u}_\Delta"] \arrow[dr, "\ulcorner_\urcorner"'] & \Sigma\,\mathsf{TM}_\Delta\,[\![\Delta]\!] \arrow[r, "\mathsf{q}_\Delta"] \arrow[d, "\mathsf{proj}"] & \mathsf{NF}_\Delta \arrow[dl, "\ulcorner_\urcorner"] \\
& \mathsf{TM}_\Delta &
\end{tikzcd}$$

**Figure 2** The type of quote and unquote for a context $\Delta$ in our proof.

In the presheaf model, the interpretation of a base type was normal forms of the base type, and the logical relation at the base type was equality of the term and the normal form. In our case, the logical predicate at the base type will say that there is a normal form which is equal to the term. The logical predicate interpretation can be seen as a dependent presheaf model.

The advantage of this approach is that it is simpler: there is no need to define presheaf models, and instead of the binary logical relation interpretation, it is enough to define the unary logical predicate interpretation. However the normalisation function now cannot be defined separately from the proof of completeness, the well-typedness of normalisation depends on the proof.

## 1.4 Structure of the proof and the paper

In this section, we give a high level sketch of the proof. Sections 3, 4, 6 are fully formalised in Agda, sections 5, 7 and 8 are partially formalised [9].

In section 2 we briefly summarize the metatheory we are working in.

In section 3 we define the syntax for type theory as a quotient inductive inductive type (QIIT) [10]. The arguments of the eliminator for the QIIT form a model of type theory.

In section 4 we define the category of renamings $\mathsf{REN}$: objects are contexts and morphisms are renamings (lists of variables).

In section 5 we define the proof-relevant Kripke logical predicate interpretation of the syntax. The interpretation has $\mathsf{REN}^{\mathsf{op}}$ as the base category and two parameters for the interpretations of $\mathsf{U}$ and $\mathsf{El}$. This interpretation can be seen as a dependent version of the presheaf model of type theory. Eg. a context in the presheaf model is interpreted as a presheaf. Now it is a family of presheaves dependent on a substitution of the appropriate types. The interpretations of base types can depend on the actual elements of the base types. The interpretation of substitutions and terms provides their fundamental theorems.

In section 6 we define neutral terms and normal forms together with their renamings and embeddings into the syntax ($\ulcorner - \urcorner$). With the help of these, we define the interpretations of $\mathsf{U}$ and $\mathsf{El}$. The interpretation of $\mathsf{U}$ at a term of type $\mathsf{U}$ will be a neutral term of type $\mathsf{U}$ which is equal to the term. Now we can interpret any term of the syntax in the logical predicate interpretation. We will denote the interpretation of a term $t$ by $[\![t]\!]$.

In section 7 we mutually define the natural transformations quote and unquote. We define them by induction on contexts and types as shown in figure 2. Quote takes a term and a semantic value at that term into a normal term and a proof that the normal term is equal to it. Unquote takes a neutral term into a semantic value at the neutral term.

Finally, in section 8, we put together the pieces by defining the normalisation function and showing that it is complete and stable. Normalisation is defined by interpreting the term in the logical predicate model at the identity semantic element and then quoting. Completeness is given by quoting as well and stability is proved by mutual induction on neutral terms and normal forms.

We conclude in section 9.

### 1.5 Related work

**TO be turned into prose!**

- Categorical approach using presheaves [6]
- Extension to System F [7,8] and coproducts [5].
- [2–4] NBE for dependent types using realizers.
- [12] NBE using adhoc formalisation, good step but not clearly related to categorical semantics.

## 2 Metatheory and notation

We are working in intensional Martin-Löf Type Theory using Agda as a vehicle [1, 15]. We extend Agda with quotient inductive inductive types (QIITs, see [10]) by using axioms. When defining an inductive type $A$, we first we declare the type by $\mathsf{data}\, A : S$ where $S$ is the sort, then we list the constructors. For inductive inductive types we first declare all the types, then following a second data keyword we list the constructors.

We also postulate functional extensionality which is a consequence of having an interval QIIT anyway. We assume $\mathsf{K}$, that is, we work in a strict type theory.

We follow Agda's convention of denoting the universe of type by $\mathsf{Set}$, we write function types as $(x : A) \to B$, implicit arguments are written by curly braces $\{x : A\} \to B$ and can be omitted or given in lower index. If some arguments are not mentioned in the context, we assume universal quantification, eg. $(y : B\, x) \to C$ means $\forall\, x.(y : B\, x) \to C$ if $x$ is not given in the context. We write $\Sigma(x : A).B$ for $\Sigma$ types.

We overload names eg. the action on objects and morphisms of a functor is denoted by the same symbol.

The identity type is denoted $- \equiv -$ and its constructor is $\mathsf{refl}$. Transport of a term through an equality is denoted ${}_{p*}a : P\,y$ if $a : P\,x$ and $p : x \equiv y$ (it is sometimes called $\mathsf{subst}$ in the literature). We denote ${}_{p*}a \equiv b$ by $a \equiv^{p} b$. We write $\mathsf{ap}$ for congruence, that is $\mathsf{ap}\,f\,p : f\,x \equiv f\,y$ if $p : x \equiv y$. For readability, we will omit transports most of the time (starting from section 5). This makes some terms non-well typed, eg. we might write $f\,a$ where $f : A \to B$ and $a : A'$ but in this case there is an equality in context which justifies $A \equiv A'$.

Sometimes we use Coq-style definitions: we write $\mathsf{d}\,(x : A) : B := t$ for defining $\mathsf{d}$ which has type $(x : A) \to B$ by $\lambda x.t$. We also use Agda-style pattern matching definitions.

## 3 Object theory

The object theory is the same[1] as in [10], we present it as a quotient inductive inductive type (QIIT). A QIIT is presented by first declaring the types that we define mutually, and then listing all the constructors.

The syntax constituting of contexts, types, substitutions and terms is declared as follows.

$$
\begin{aligned}
&\mathsf{data\,Con} &&: \mathsf{Set}\\
&\mathsf{data\,Ty} &&: \mathsf{Con} \to \mathsf{Set}\\
&\mathsf{data\,Tms} &&: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}\\
&\mathsf{data\,Tm} &&: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set}
\end{aligned}
$$

We use the convention of naming contexts $\Gamma, \Delta, \Theta$, types $A, B$, terms $t, u$, substitutions $\sigma, \nu, \delta$.

We define a basic type theory with an uninterpreted base type $\mathsf{U}$, a family over this type $\mathsf{El}$ and dependent function space $\Pi$ with constructor $\mathsf{lam}$ and eliminator $\mathsf{app}$. Our type theory is given as explicit substitution calculus, hence we the need constructors $-[-]$ for substituted types and terms. The constructors can be summarized as follows.

- Substitutions form a category with a terminal object. This includes the categorical substitution laws for types $[\mathsf{id}]$ and $[][]$.
- Substitution laws for types $\mathsf{U}[]$, $\mathsf{El}[]$, $\Pi[]$.
- The laws of comprehension which state that we have the natural isomorphism

  $$\pi_1\beta, \pi_2\beta \circlearrowleft \qquad -,- \downarrow \frac{\sigma : \mathsf{Tms}\,\Gamma\,\Delta \qquad \mathsf{Tm}\,\Gamma\,A[\sigma]}{\mathsf{Tms}\,\Gamma\,(\Delta, A)} \uparrow \pi_1, \pi_2 \qquad \curvearrowright \pi\eta$$

  where naturality[2] is given by $,\circ$.
- The laws for function space which are given by the natural isomorphism

  $$\Pi\beta \circlearrowleft \qquad \mathsf{lam} \downarrow \frac{\mathsf{Tm}\,(\Gamma, A)\,B}{\mathsf{Tm}\,\Gamma\,(\Pi\,A\,B)} \uparrow \mathsf{app} \qquad \curvearrowright \Pi\eta$$

  where naturality is given by $\mathsf{lam}[]$.

---

[1] Steven Schäfer pointed us to [16] which shows that in the presentation [10] the equalities $[\mathsf{id}]\mathsf{t}$ and $[][]\mathsf{t}$ (identity and associativity laws of term substitution) can be derived from the others. This is why we omitted these equalities from this presentation and the formal development.

[2] If one direction of an isomorphism is natural, so is the other. This is why it is enough to state naturality for $-,-$ and not for $\pi_1$, $\pi_2$.

We list the point constructors in the left column and the equality constructors in the right column.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{data}} \\
\quad \cdot & : \mathsf{Con} \\
\quad -,- & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
\quad -[-] & : \mathsf{Ty}\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma \\
\quad \mathsf{U} & : \mathsf{Ty}\,\Gamma \\
\quad \mathsf{El} & : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma \\
\quad \Pi & : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma, A) \to \mathsf{Ty}\,\Gamma \\
\quad \mathsf{id} & : \mathsf{Tms}\,\Gamma\,\Gamma \\
\quad - \circ - & : \mathsf{Tms}\,\Theta\,\Delta \to \mathsf{Tms}\,\Gamma\,\Theta \to \mathsf{Tms}\,\Gamma\,\Delta \\
\quad \epsilon & : \mathsf{Tms}\,\Gamma\,\cdot \\
\quad -,- & : (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \to \mathsf{Tms}\,\Gamma\,(\Delta, A) \\
\quad \pi_1 & : \mathsf{Tms}\,\Gamma\,(\Delta, A) \to \mathsf{Tms}\,\Gamma\,\Delta \\
\quad -[-] & : \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \\
\quad \pi_2 & : (\sigma : \mathsf{Tms}\,\Gamma\,(\Delta, A)) \to \mathsf{Tm}\,\Gamma\,A[\pi_1\,\sigma] \\
\quad \mathsf{lam} & : \mathsf{Tm}\,(\Gamma, A)\,B \to \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \\
\quad \mathsf{app} & : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to \mathsf{Tm}\,(\Gamma, A)\,B
\end{array}
$$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{data}} \\
\quad [\mathsf{id}] & : A[\mathsf{id}] \equiv A \\
\quad [][] & : A[\sigma][\nu] \equiv A[\sigma \circ \nu] \\
\quad \mathsf{U}[] & : \mathsf{U}[\sigma] \equiv \mathsf{U} \\
\quad \mathsf{El}[] & : (\mathsf{El}\,\hat{A})[\sigma] \equiv \mathsf{El}\,({}_{\mathsf{U}[]*}\hat{A}[\sigma]) \\
\quad \Pi[] & : (\Pi\,A\,B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^A]) \\
\quad \mathsf{id}\circ & : \mathsf{id} \circ \sigma \equiv \sigma \\
\quad \circ\mathsf{id} & : \sigma \circ \mathsf{id} \equiv \sigma \\
\quad \circ\circ & : (\sigma \circ \nu) \circ \delta \equiv \sigma \circ (\nu \circ \delta) \\
\quad \epsilon\eta & : \{\sigma : \mathsf{Tms}\,\Gamma\,\cdot\} \to \sigma \equiv \epsilon \\
\quad \pi_1\beta & : \pi_1\,(\sigma, t) \equiv \sigma \\
\quad \pi\eta & : (\pi_1\,\sigma, \pi_2\,\sigma) \equiv \sigma \\
\quad ,\circ & : (\sigma, t) \circ \nu \equiv (\sigma \circ \nu), ({}_{[][]*}t[\nu]) \\
\quad \pi_2\beta & : \pi_2\,(\sigma, t) \equiv^{\pi_1\beta} t \\
\quad \Pi\beta & : \mathsf{app}\,(\mathsf{lam}\,t) \equiv t \\
\quad \Pi\eta & : \mathsf{lam}\,(\mathsf{app}\,t) \equiv t \\
\quad \mathsf{lam}[] & : (\mathsf{lam}\,t)[\sigma] \equiv^{\Pi[]} \mathsf{lam}\,(t[\sigma^A])
\end{array}
$$

Note that the equality $\pi_2\beta$ lives over $\pi_1\beta$. Also, we had to use ${}_{-*}-$ to typecheck $\mathsf{El}[]$ and $,\circ$. We used lifting of a substitution in the types of $\Pi[]$ and $\mathsf{lam}[]$. It is defined as follows.

$$(\sigma : \mathsf{Tms}\,\Gamma\,\Delta)^A : \mathsf{Tms}\,(\Gamma, A[\sigma])\,(\Delta, A) := (\sigma \circ \pi_1\,\mathsf{id}), ({}_{[][]*}\pi_2\,\mathsf{id})$$

We use the categorical $\mathsf{app}$ operator but the usual one $(-\$-)$ can also be derived.

$$
\begin{array}{ll}
< (u : \mathsf{Tm}\,\Gamma\,A) > & : \mathsf{Tms}\,\Gamma\,(\Gamma, A) := \mathsf{id}, {}_{[\mathsf{id}]^{-1}*}u \\
(t : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B))\$(u : \mathsf{Tm}\,\Gamma\,A) : B[< u >] & := (\mathsf{app}\,t)[< u >]
\end{array}
$$

When we define a function from the above syntax, we need to use the eliminator. The eliminator has 4 motives corresponding to what Con, Ty, Tms and Tm get mapped to and one method for each constructor including the equality constructors. The methods for point constructors are elements of the motives to which the constructor is mapped. The methods for the equality constructors demonstrate soundness, that is, the semantic constructions respect the syntactic equalities. The eliminator comes in two different flavours: the non-dependent and dependent version. In our constructions we use the dependent version. The motives and methods for the non-dependent eliminator (recursor) collected together form a model of type theory, they are basically the same[3] as Dubjer's Categories with Families [13].

As an example we list the motives and a few methods of the dependent eliminator. An algorithm for deriving them is given in [10]. As names we use the names of the constructors

[3] Dybjer uses the usual application operator, we use the categorical one, the projections $\pi_1$, $\pi_2$ are defined differently and Dybjer lists some equations derivable from the others, we omit these. However all the operators and the laws are inter-derivable.

followed by an upper index $\mathsf{M}$.

$$
\begin{aligned}
&\mathsf{Con}^{\mathsf{M}} && : \mathsf{Con} \to \mathsf{Set} \\
&\mathsf{Ty}^{\mathsf{M}} && : (\mathsf{Con}^{\mathsf{M}}\,\Gamma) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
&\mathsf{Tms}^{\mathsf{M}} && : (\mathsf{Con}^{\mathsf{M}}\,\Gamma) \to (\mathsf{Con}^{\mathsf{M}}\,\Delta) \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Set} \\
&\mathsf{Ty}^{\mathsf{M}} && : (\Gamma^{\mathsf{M}} : \mathsf{Con}^{\mathsf{M}}\,\Gamma) \to \mathsf{Ty}^{\mathsf{M}}\,\Gamma^{\mathsf{M}}\,A \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Set} \\
&\mathsf{id}^{\mathsf{M}} && : \mathsf{Tms}^{\mathsf{M}}\,\Gamma^{\mathsf{M}}\,\Gamma^{\mathsf{M}}\,\mathsf{id} \\
&- \circ^{\mathsf{M}} - && : \mathsf{Tms}^{\mathsf{M}}\,\Theta^{\mathsf{M}}\,\Delta^{\mathsf{M}}\,\sigma \to \mathsf{Tms}^{\mathsf{M}}\,\Gamma^{\mathsf{M}}\,\Theta^{\mathsf{M}}\,\nu \to \mathsf{Tms}^{\mathsf{M}}\,\Gamma^{\mathsf{M}}\,\Delta^{\mathsf{M}}\,(\sigma \circ \nu) \\
&\circ\mathsf{id}^{\mathsf{M}} && : \sigma^{\mathsf{M}} \circ^{\mathsf{M}} \mathsf{id}^{\mathsf{M}} \equiv^{\circ\mathsf{id}} \sigma^{\mathsf{M}} \\
&\pi_2\beta^{\mathsf{M}} && : \pi_2^{\mathsf{M}}\,(\rho^{\mathsf{M}},^{\mathsf{M}} t^{\mathsf{M}}) \equiv^{\pi_1\beta^{\mathsf{M}},\pi_2\beta}\, t^{\mathsf{M}}
\end{aligned}
$$

Note that the method equality $\circ\mathsf{id}^{\mathsf{M}}$ lives over the constructor $\circ\mathsf{id}$ while the method equality $\pi_2\beta^{\mathsf{M}}$ lives both over the method equality $\pi_1\beta^{\mathsf{M}}$ and the constructor $\pi_2\beta$. Later, for brevity, we will omit these equality dependencies and the usages of $\_{*}-$ as well.

## 4 The category of renamings

In this section we define the category of renamings $\mathsf{REN}$. Objects in this category are contexts, morphisms are renamings: lists of de Bruijn variables. In a syntax with variables names, the first two examples below are renamings, while the third one is not.

$$
\begin{aligned}
&(i \mapsto y, j \mapsto x, k \mapsto x) && : \mathsf{Vars}\,(x : A, y : B)\,(i : B, j : A, k : A) \\
&(i \mapsto z, j \mapsto x) && : \mathsf{Vars}\,(x : A \to B, y : A, z : B)\,(i : B, j : A \to B) \\
&(i \mapsto x\$y, j \mapsto x) && : \mathsf{Vars}\,(x : A \to B, y : A, z : B)\,(i : B, j : A \to B)
\end{aligned}
$$

We define the inductive type of variables $\mathsf{Var}$ and renamings $\mathsf{Vars}$ together with their embeddings into substitutions. This is an inductive-recursive definition as $\ulcorner - \urcorner$ for renamings needs to be defined mutually with renamings.

$$
\begin{aligned}
&\mathsf{data}\,\mathsf{Var} && : (\Psi : \mathsf{Con}) \to \mathsf{Ty}\,\Psi \to \mathsf{Set} \\
&\quad \mathsf{vze} && : \mathsf{Var}\,(\Psi, A)\,(A[\pi_1\,\mathsf{id}]) \\
&\quad \mathsf{vsu} && : \mathsf{Var}\,\Psi\,A \to \mathsf{Var}\,(\Psi, B)(A[\pi_1\,\mathsf{id}]) \\
&\ulcorner - \urcorner && : \mathsf{Vars}\,\Psi\,\Omega \to \mathsf{Tms}\,\Psi\,\Omega \\
&\mathsf{data}\,\mathsf{Vars} && : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
&\quad \epsilon && : \mathsf{Vars}\,\Psi\,\cdot \\
&\quad -,- && : (\beta : \mathsf{Vars}\,\Psi\,\Omega) \to \mathsf{Var}\,\Psi\,A[\ulcorner\beta\urcorner] \to \mathsf{Vars}\,\Psi\,(\Omega, A) \\
&\ulcorner - \urcorner && : \mathsf{Var}\,\Psi\,A \to \mathsf{Tm}\,\Psi\,A \\
&\ulcorner \mathsf{vze} \urcorner && := \pi_2\,\mathsf{id} \\
&\ulcorner \mathsf{vsu}\,x \urcorner && := \ulcorner x \urcorner[\pi_1\,\mathsf{id}] \\
&\ulcorner \epsilon \urcorner && := \epsilon \\
&\ulcorner \beta, x \urcorner && := \ulcorner \beta \urcorner, \ulcorner x \urcorner
\end{aligned}
$$

Variables are typed de Bruijn indices. $\mathsf{vze}$ projects out the last element of the context, $\mathsf{vsu}$ extends the context, and the type $A : \mathsf{Ty}\,\Psi$ needs to be weekened in both cases because we need to interpret it in $\Psi$ extended by another type. Renamings are lists of variables with the appropriate types. Embedding of variables into terms uses the projections and the identity substitution, and embedding renamings is pointwise.

We use the naming convention $\Psi, \Omega, \Xi$ for objects of $\mathsf{REN}$, $x, y$ for variables, $\beta, \gamma, \iota$ for renamings.

We need identity and composition of renamings for the categorical structure. To define them, we also need weakening and renaming of variables together with laws relating their embeddings to terms. We only list the types as the definitions are straightforward inductions. The only catch is that the identity substitution uses weakening and not the other way.

$$
\begin{array}{llll}
\mathsf{wk} & : \mathsf{Vars}\,\Psi\,\Omega \to \mathsf{Vars}\,(\Psi, A)\,\Omega & \ulcorner\mathsf{wk}\urcorner & : \ulcorner\beta\urcorner \circ \pi_1 \mathsf{id} \equiv \ulcorner\mathsf{wk}\,\beta\urcorner \\
\mathsf{id} & : \mathsf{Vars}\,\Psi\,\Psi & \ulcorner\mathsf{id}\urcorner & : \ulcorner\mathsf{id}\urcorner \equiv \mathsf{id} \\
- \circ - & : \mathsf{Vars}\,\Xi\,\Omega \to \mathsf{Vars}\,\Psi\,\Xi \to \mathsf{Vars}\,\Psi\,\Omega & \ulcorner\circ\urcorner & : \ulcorner\beta\urcorner \circ \ulcorner\gamma\urcorner \equiv \ulcorner\beta \circ \gamma\urcorner \\
-[-] & : \mathsf{Var}\,\Omega\,A \to (\beta : \mathsf{Vars}\,\Psi\,\Omega) \to \mathsf{Var}\,\Psi\,A[\ulcorner\beta\urcorner] & \ulcorner[]\urcorner & : \ulcorner x\urcorner[\ulcorner\beta\urcorner] \equiv \ulcorner x[\beta]\urcorner
\end{array}
$$

Renamings form a category, we omit the statement and proofs of the categorical laws.

## 5 The logical predicate interpretation

In this section, after defining a few categorical notions, we define the proof-relevant Kripke logical predicate interpretation of the type theory given in section 3. It can also be seen as a dependent version of the presheaf model of type theory [14].

A contravariant presheaf over a category $\mathcal{C}$ is denoted $\Gamma : \mathsf{PSh}\,\mathcal{C}^{\mathsf{op}}$. It is given by the following data: given $I : |\mathcal{C}|$, a set $\Gamma\,I$, and given $f : \mathcal{C}(J, I)$ a function $\Gamma\,f : \Gamma\,I \to \Gamma\,J$. Moreover, we have $\mathsf{idP}\,\Gamma : \Gamma\,\mathsf{id}\,\alpha \equiv \alpha$ and $\mathsf{compP}\,\Gamma : \Gamma\,(f \circ g)\,\alpha \equiv \Gamma\,g\,(\Gamma\,f\,\alpha)$ for $\alpha : \Gamma\,I$, $f : \mathcal{C}(J, I)$, $g : \mathcal{C}(K, J)$.

Given $\Gamma : \mathsf{PSh}\,\mathcal{C}$, a family of presheaves over $\Gamma$ is denoted $A : \mathsf{FamPSh}\,\Gamma$. It is given by the following data: given $\alpha : \Gamma\,I$, a set $A_I\,\alpha$ and given $f : \mathcal{C}(J, I)$, a function $A\,f : A_I\,\alpha \to A_J\,(\Gamma\,f\,\alpha)$. In addition, we have the functor laws $\mathsf{idF}\,A : A\,\mathsf{id}\,v \equiv^{\mathsf{idP}} v$ and $\mathsf{compF}\,A : A\,(f \circ g)\,v \equiv^{\mathsf{compP}} A\,g\,(A\,f\,v)$ for $\alpha : \Gamma\,I$, $v : A\,\alpha$, $f : \mathcal{C}(J, I)$, $g : \mathcal{C}(K, J)$.

A natural transformation between presheaves $\Gamma$ and $\Delta$ is denoted $\sigma : \Gamma \dot{\to} \Delta$. It is given by a function $\sigma : \{I : |\mathcal{C}|\} \to \Gamma\,I \to \Delta\,I$ together with the condition $\mathsf{natn}\,\sigma : \Delta\,f\,(\sigma_I\,\alpha) \equiv \sigma_J\,(\Gamma\,f\,\alpha)$ for $\alpha : \Gamma\,I$, $f : \mathcal{C}(J, I)$.

A section[4] from a presheaf $\Gamma$ to a family of presheaves $A$ over $\Gamma$ is denoted $t : \Gamma \xrightarrow{\mathsf{s}} A$. It is given by a function $t : \{I : |\mathcal{C}|\} \to (\alpha : \Gamma\,I) \to A_I\,\alpha$ together with the naturality condition $\mathsf{natS}\,t\,\alpha\,f : A\,f\,(t\,\alpha) \equiv t\,(\Gamma\,f\,\alpha)$ for $f : \mathcal{C}(J, I)$.

Given $\Gamma : \mathsf{PSh}\,\mathcal{C}^{\mathsf{op}}$ and $A : \mathsf{FamPSh}\,\Gamma$ we can define $\Sigma\,\Gamma\,A : \mathsf{PSh}\,\mathcal{C}^{\mathsf{op}}$ by $(\Sigma\,\Gamma\,A)\,I := \Sigma(\alpha : \Gamma\,I).A_i\,\alpha$ and $(\Sigma\,\Gamma\,A)\,f\,(\alpha, a) := (\Gamma\,f\,\alpha, A\,f\,a)$.

Given $\sigma : \Gamma \dot{\to} \Delta$ and $A : \mathsf{FamPSh}\,\Delta$, we define $A[\sigma] : \mathsf{FamPSh}\,\Gamma$ by $A[\sigma]_I\,\alpha := A_I\,(\sigma_I\,\alpha)$ and $A[\sigma]\,f\,\alpha := {}_{\mathsf{natn}\,\sigma *}A\,f\,\alpha$ for $\alpha : \Gamma\,I$ and $f : \mathcal{C}(J, I)$.

The weakening natural transformation $\mathsf{wk} : \Sigma\,\Gamma\,A \dot{\to} \Gamma$ is defined by $\mathsf{wk}_I\,(\alpha, a) := \alpha$.

Lifting of a section $t : \Gamma \xrightarrow{\mathsf{s}} A$ by a family of presheaves $B : \mathsf{FamPSh}\,\Gamma$ is a natural transformation $t^B : \Sigma\,\Gamma\,B \dot{\to} \Sigma\,(\Sigma\,(\Gamma\,A))\,B[\mathsf{wk}]$. It is defined as ${t^B}_I\,(\alpha, b) := (\alpha, t_I\,\alpha, b)$.

To define the logical predicate interpretation of the syntax, we need to give the motives and methods for the eliminator. We will denote the interpretation of a syntactic construct $t$

---

[4] $t : \Gamma \xrightarrow{\mathsf{s}} A$ is called a section because it can be viewed as a section of the first projection from $\Sigma\,\Gamma\,A$ to $\Gamma$ but we define it without using the projection.

by $[\![t]\!]$. The following table gives the motives of the eliminator.

$$
\begin{array}{llll}
\Gamma : \mathsf{Con} & \mathsf{TM}_\Gamma \;= \mathsf{Tms} - \Gamma & : \mathsf{PSh}\,\mathsf{REN}^{\mathsf{op}} & [\![\Gamma]\!] \;: \mathsf{FamPSh}\,\mathsf{TM}_\Gamma \\
A : \mathsf{Ty}\,\Gamma & \mathsf{TM}_A = \mathsf{Tm} - A[-] & : \mathsf{FamPSh}\,\mathsf{TM}_\Gamma & [\![A]\!] : \mathsf{FamPSh}\left(\Sigma\left(\Sigma\left(\mathsf{TM}_\Gamma\,\mathsf{TM}_A\right)\right)[\![\Gamma]\!][\mathsf{wk}]\right) \\
\sigma : \mathsf{Tms}\,\Gamma\,\Delta & \mathsf{TM}_\sigma \;= (\sigma \circ -) & : \mathsf{TM}_\Gamma \dot{\to} \mathsf{TM}_\Delta & [\![\sigma]\!] \;: \Sigma\,\mathsf{TM}_\Gamma\,[\![\Gamma]\!] \xrightarrow{\mathsf{s}} [\![\Delta]\!][\mathsf{TM}_\sigma][\mathsf{wk}] \\
t : \mathsf{Tm}\,\Gamma\,A & \mathsf{TM}_t \;= t[-] & : \mathsf{TM}_\Gamma \xrightarrow{\mathsf{s}} \mathsf{TM}_A & [\![t]\!] \;\;: \Sigma\,\mathsf{TM}_\Gamma\,[\![\Gamma]\!] \xrightarrow{\mathsf{s}} [\![A]\!][\mathsf{TM}_t{}^{[\![\Gamma]\!]}]
\end{array}
$$

First we define the syntactic presheaf interpretation $\mathsf{TM}$ as given in the table. $\mathsf{TM}_\Delta$ is a presheaf over $\mathsf{REN}^{\mathsf{op}}$, the action on morphisms is $\mathsf{TM}_\Delta\,(\beta : \mathsf{Vars}\,\Omega\,\Psi)\,\sigma := \sigma \circ \ulcorner\beta\urcorner$. For a type $\Gamma \vdash A$, $\mathsf{TM}_A$ is a family of presheaves over $\mathsf{TM}_\Gamma$, substitutions are natural transformations, terms are sections. The action on morphisms for $\mathsf{TM}_A$ and the laws are straightforward. $\mathsf{TM}$ is not a presheaf model, it is just the syntax in a different structure so that it matches the motives of a presheaf model.

A context will be mapped to a family of presheaves over $\mathsf{TM}_\Delta$. That is, for every substitution $\rho : \mathsf{TM}_\Delta\,\Psi$ we have a set $[\![\Delta]\!]_\Psi\,\rho$ which expresses that the logical predicate holds for $\rho$. Moreover, we have the renaming $[\![\Gamma]\!]\,\beta : [\![\Gamma]\!]\,\rho \to [\![\Gamma]\!]\,(\mathsf{TM}_\Gamma\,\beta\,\rho)$.

$[\![A]\!]$ is the logical predicate at a type $A$. It depends on a substitution (for which the predicate needs to hold) and a term. $[\![A]\!]_\Psi\,(\rho, s, \alpha)$ expresses that the logical predicate holds for term $s : \mathsf{Tm}\,\Psi\,A[\rho]$. It is also stable under renamings.

$$
\frac{A : \mathsf{Ty}\,\Gamma \qquad \Psi : |\mathsf{REN}^{\mathsf{op}}| \qquad \rho : \mathsf{TM}_\Gamma\,\Psi \qquad s : \mathsf{TM}_A\,\rho \qquad \alpha : [\![\Gamma]\!]_\Psi\,\rho}{[\![A]\!]_\Psi\,(\rho, s, \alpha) : \mathsf{Set}}
$$

$$[\![A]\!]\,\beta : [\![A]\!]\,(\rho, s, \alpha) \to [\![A]\!]\,(\mathsf{TM}_\Gamma\,\beta\,\rho, \mathsf{TM}_A\,\beta\,s, [\![\Gamma]\!]\,\beta\,\alpha)$$

A substitution $\sigma$ is mapped to $[\![\sigma]\!]$ which expresses the fundamental theorem of the logical predicate for $\sigma$: for any other substitution $\rho$ for which the predicate holds, we can compose it with $\sigma$ and the predicate will hold for the composition. The fundamental theorem is also natural.

$$
\frac{\sigma : \mathsf{Tms}\,\Gamma\,\Delta \qquad \Psi : |\mathsf{REN}^{\mathsf{op}}| \qquad \rho : \mathsf{TM}_\Gamma\,\Psi \qquad \alpha : [\![\Gamma]\!]_\Psi\,\rho}{[\![\sigma]\!]_\Psi\,(\rho, \alpha) : [\![\Delta]\!]_\Psi\,(\sigma \circ \rho)}
$$

$$[\![\Delta]\!]\,\beta\,([\![\sigma]\!]\,(\rho, \alpha)) \equiv [\![\sigma]\!]\,(\mathsf{TM}_\Gamma\,\beta\,\rho, [\![\Gamma]\!]\,\beta\,\alpha)$$

A term $t$ is mapped to the fundamental theorem for the term: given a substitution $\rho$ for which the predicate holds, it also holds for $t[\rho]$ in a natural way.

$$
\frac{t : \mathsf{Tm}\,\Gamma\,A \qquad \Psi : |\mathsf{REN}^{\mathsf{op}}| \qquad \rho : \mathsf{TM}_\Gamma\,\Psi \qquad \alpha : [\![\Gamma]\!]_\Psi\,\rho}{[\![t]\!]_\Psi\,(\rho, \alpha) : [\![A]\!]_\Psi\,(\rho, t[\rho], \alpha)}
$$

$$[\![A]\!]\,\beta\,([\![t]\!]\,(\rho, \alpha)) \equiv [\![t]\!]\,(\mathsf{TM}_\Gamma\,\beta\,\rho, [\![\Gamma]\!]\,\beta\,\alpha)$$

We define the presheaf $\mathsf{TM}^{\mathsf{U}} : \mathsf{PSh}\,\mathsf{REN}^{\mathsf{op}}$ and a family over it $\mathsf{TM}^{\mathsf{El}} : \mathsf{FamPSh}\,\mathsf{TM}^{\mathsf{U}}$. The actions on objects are $\mathsf{TM}^{\mathsf{U}}\,\Psi := \mathsf{Tm}\,\Psi\,\mathsf{U}$ and $\mathsf{TM}^{\mathsf{El}}{}_\Psi\,\hat{A} := \mathsf{Tm}\,\Psi\,(\mathsf{El}\,\hat{A})$. The actions on a morphism $\beta$ is just substitution $-[\ulcorner\beta\urcorner]$.

Note that the base category of the logical predicate interpretation is fixed to $\mathsf{REN}^{\mathsf{op}}$. However we parameterise the interpretation by the predicate at the base type $\mathsf{U}$ and base family $\mathsf{El}$. These are denoted by $\bar{\mathsf{U}}$ and $\bar{\mathsf{El}}$ and have the following types.

$$\bar{\mathsf{U}} \;: \mathsf{FamPSh}\,\mathsf{TM}^{\mathsf{U}}$$

$$\bar{\mathsf{El}} : \mathsf{FamPSh}\left(\Sigma\left(\Sigma\left(\mathsf{TM}^{\mathsf{U}}\,\mathsf{TM}^{\mathsf{El}}\right)\right)\bar{\mathsf{U}}[\mathsf{wk}]\right)$$

Now we list the methods for each constructor in the same order as we have given them in section 3. We omit the proofs of functoriality/naturality only for reasons of space.

The logical predicate trivially holds at the empty context, and it holds at an extended context for $\rho$ if it holds at the smaller context at $\pi_1\,\rho$ and if it holds at the type which extends the context for $\pi_2\,\rho$. The second part obviously depends on the first. The action on morphisms for context extension is pointwise. Here we omitted some usages of $\_{}_*\_$ eg. $[\![\Gamma]\!]\,\beta\,\alpha$ is only well-typed in that position when we transport along the equality $\pi_1\,\rho \circ \ulcorner\beta\urcorner \equiv \pi_1\,(\rho \circ \ulcorner\beta\urcorner)$. From now on we will omit transports and the usages of $\ulcorner - \urcorner$ in most cases for readability.

$$
\begin{aligned}
&[\![\cdot]\!]_\Psi\,(\rho : \mathsf{TM}.\,\Psi) && := \top\\
&[\![\Gamma, A]\!]_\Psi\,(\rho : \mathsf{TM}_{\Gamma,A}\,\Psi) && := \Sigma(\alpha : [\![\Gamma]\!]_\Psi\,(\pi_1\,\rho)).[\![A]\!]_\Psi\,(\pi_1\,\rho, \pi_2\,\rho, \alpha)\\
&[\![\Gamma, A]\!]\,(\beta : \mathsf{Vars}\,\Omega\,\Psi)\,(\alpha, a) && := ([\![\Gamma]\!]\,\beta\,\alpha, [\![A]\!]\,\beta\,a)
\end{aligned}
$$

The logical predicate at a substituted type is the logical predicate at the type and we need to use the fundamental theorem at the substitution to lift the witness of the predicate for the substitution. Renaming a substituted type is the same as renaming in the original type. The logical predicate at the base type and family says what we have given as parameters. Renaming also comes from these parameters.

$$
\begin{aligned}
&[\![A[\sigma]]\!]\,(\rho, s, \alpha) := [\![A]\!]\,(\sigma \circ \rho, s, [\![\sigma]\!]\,(\rho, \alpha)) && [\![A[\sigma]]\!]\,\beta\,a := [\![A]\!]\,\beta\,a\\
&[\![\mathsf{U}]\!]\,(\rho, s, \alpha) \quad := \bar{\mathsf{U}}\,({\mathsf{U}[]_*}s) && [\![\mathsf{U}]\!]\,\beta\,a \quad := \bar{\mathsf{U}}\,\beta\,a\\
&[\![\mathsf{El}\,\hat{A}]\!]\,(\rho, s, \alpha) := \bar{\mathsf{El}}\,(\hat{A}[\rho], s, [\![\hat{A}]\!]\,(\rho, \alpha)) && [\![\mathsf{El}\,\hat{A}]\!]\,\beta\,a := \bar{\mathsf{El}}\,\beta\,a
\end{aligned}
$$

The logical predicate holds for a function $s$ when we have that if the predicate holds for an argument $u$ (at $A$, witnessed by $v$), so it holds for $s\$u$ at $B$. In addition, we have a Kripke style generalisation: this should be true for $s[\beta]$ given a morphism $\beta$ in a natural way. Renaming a witness of the logical predicate at the function type is postcomposing the Kripke morphism by it.

$$
\begin{aligned}
&[\![\Pi\,A\,B]\!]_\Psi\,(\rho : \mathsf{TM}_\Gamma\,\Psi, s, \alpha)\\
&\qquad := \Sigma\Big(\mathsf{map} : \big(\beta : \mathsf{Vars}\,\Omega\,\Psi\big)\big(u : \mathsf{TM}_A\,(\rho\circ\beta)\big)\big(v : [\![A]\!]_\Omega\,(\rho\circ\beta, u, [\![\Gamma]\!]\,\beta\,\alpha)\big)\\
&\qquad\qquad\qquad \to [\![B]\!]_\Omega\,\big((\rho\circ\beta, u), s[\beta]\$u, ([\![\Gamma]\!]\,\beta\,\alpha, v)\big)\Big)\\
&\qquad\qquad .\forall\beta, u, v, \gamma.[\![B]\!]\,\gamma\,(\mathsf{map}\,\beta\,u\,v) \equiv \mathsf{map}\,(\beta\circ\gamma)\,(u[\gamma])\,([\![A]\!]\,\gamma\,v)\\
&[\![\Pi\,A\,B]\!]\,\beta'\,(\mathsf{map}, \mathsf{nat}) := (\lambda\beta.\mathsf{map}\,(\beta'\circ\beta), \lambda\beta.\mathsf{nat}\,(\beta'\circ\beta))
\end{aligned}
$$

Now we list the methods for substitution constructors, that is, we prove the fundamental theorem for substitutions. We omit the naturality proofs. Here object theoretic constructs map to their metatheoretic counterparts: identity becomes identity, composition becomes composition, the empty substitution becomes the element of the unit type, comprehension becomes pairing, first projection becomes first projection.

$$
\begin{aligned}
&[\![\mathsf{id}]\!]\,(\rho, \alpha) && := \alpha\\
&[\![\sigma\circ\nu]\!]\,(\rho, \alpha) && := [\![\sigma]\!]\,(\nu\circ\rho, [\![\nu]\!]\,(\rho, \alpha))\\
&[\![\epsilon]\!]\,(\rho, \alpha) && := \mathsf{tt}\\
&[\![\sigma, t]\!]\,(\rho, \alpha) && := [\![\sigma]\!]\,(\rho, \alpha), [\![t]\!]\,(\rho, \alpha)\\
&[\![\pi_1\,\sigma]\!]\,(\rho, \alpha) && := \mathsf{proj}_1\,([\![\sigma]\!]\,(\rho, \alpha))
\end{aligned}
$$

The fundamental theorem for substituted terms and the second projection is again just composition and second projection.

$$
\begin{aligned}
&[\![t[\sigma]]\!]\,(\rho,\alpha)\ := [\![t]\!]\,(\sigma\circ\rho,[\![\sigma]\!]\,(\rho,\alpha))\\
&[\![\pi_2\,\sigma]\!]\,(\rho,\alpha) := \mathsf{proj}_2\,([\![\sigma]\!]\,(\rho,\alpha))
\end{aligned}
$$

The fundamental theorems for $\mathsf{lam}$ and $\mathsf{app}$ are more interesting. For $\mathsf{lam}$, the $\mathsf{map}$ function is using the fundamental theorem for $t$ which is in the context extended by the domain type $A : \mathsf{Ty}\,\Gamma$, so we need to supply an extended substitution and a witness of the predicate. Moreover, we need to rename the substitution $\rho$ and the witness of the predicate $\alpha$ to account for the Kripke property. The naturality is given by the naturality of the term itself. Application uses the $\mathsf{map}$ part of the logical predicate and the identity renaming.

$$
\begin{aligned}
[\![\mathsf{lam}\,t]\!]\,(\rho,\alpha) := \Big(&\lambda\beta,u,v.[\![t]\!]\,\big((\rho\circ\beta,u),([\![\Gamma]\!]\,\beta\,\alpha,v)\big)\\
&,\lambda\beta,u,v,\gamma.\mathsf{natS}\,[\![t]\!]\,\big((\rho\circ\beta,u),([\![\Gamma]\!]\,\beta\,\alpha,v)\big)\,\gamma\Big)\\
[\![\mathsf{app}\,t]\!]\,(\rho,\alpha) := \mathsf{map}\,&\big([\![t]\!]\,(\pi_1\,\rho,\mathsf{proj}_1\,\alpha)\big)\,\mathsf{id}\,(\pi_2\,\rho)\,(\mathsf{proj}_2\,\alpha)
\end{aligned}
$$

Finally, we need to provide methods for the equality constructors. As our metatheory uses $\mathsf{K}$, to give an equality between two families of presheaves, it is enough to show that the action on objects and morphisms coincide, there is no need to check equality of witnesses of the laws, and similarly for sections. We won't list these proofs here as they are all straightforward.

## 6 Normal forms

In this section we define the presheaves and families of neutral terms and normal forms.

We define $\eta$-long $\beta$-normal forms mutually with neutral terms. Neutral terms are terms where a variable is in a key position which precludes the application of the rule $\Pi\beta$. Embeddings back into the syntax are defined mutually in the obvious way. Note that neutral terms and normal forms are indexed by types, not normal types.

$$
\begin{aligned}
&\mathsf{data\ Ne} &&: (\Gamma:\mathsf{Con})\to\mathsf{Ty}\,\Gamma\to\mathsf{Set}\\
&\mathsf{data\ Nf} &&: (\Gamma:\mathsf{Con})\to\mathsf{Ty}\,\Gamma\to\mathsf{Set}\\
&\ulcorner{-}\urcorner &&: \mathsf{Nf}\,\Gamma\,A\to\mathsf{Tm}\,\Gamma\,A\\
&\mathsf{data\ Ne}\\
&\quad\mathsf{var} &&: \mathsf{Var}\,\Gamma\,A\to\mathsf{Ne}\,\Gamma\,A\\
&\quad\mathsf{app} &&: \mathsf{Ne}\,\Gamma\,(\Pi\,A\,B)\to(v:\mathsf{Nf}\,\Gamma\,A)\to\mathsf{Ne}\,\Gamma\,(B[<\ulcorner v\urcorner>])\\
&\mathsf{data\ Nf}\\
&\quad\mathsf{neuU} &&: \mathsf{Ne}\,\Gamma\,\mathsf{U}\to\mathsf{Nf}\,\Gamma\,\mathsf{U}\\
&\quad\mathsf{neuEl} &&: \mathsf{Ne}\,\Gamma\,(\mathsf{El}\,\hat{A})\to\mathsf{Nf}\,\Gamma\,(\mathsf{El}\,\hat{A})\\
&\quad\mathsf{lam} &&: \mathsf{Nf}\,(\Gamma,A)\,B\to\mathsf{Nf}\,\Gamma\,(\Pi\,A\,B)\\
&\ulcorner{-}\urcorner &&: \mathsf{Ne}\,\Gamma\,A\to\mathsf{Tm}\,\Gamma\,A
\end{aligned}
$$

We define lists of neutral terms and normal forms. $X$ is a parameter of the list, it can stand for both Ne and Nfs.

$$
\begin{aligned}
&\mathsf{data}\ {-}\mathsf{s}\,(X : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set}) : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}\\
&\ulcorner{-}\urcorner : X\mathsf{s}\,\Gamma\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta\\
&\mathsf{data}\ X\mathsf{s}\\
&\quad \epsilon \quad\ \, : X\mathsf{s}\,\Gamma\,\cdot\\
&\quad {-},{-} : (\tau : X\mathsf{s}\,\Gamma\,\Delta) \to X\,\Gamma\,A[\ulcorner\tau\urcorner] \to X\mathsf{s}\,\Gamma\,(\Delta, A)
\end{aligned}
$$

We also need renamings of (lists of) normal forms and neutral terms together with lemmas relating their embeddings to terms. Again, $X$ can stand for both Ne and Nf.

$$
\begin{aligned}
&{-}[{-}]\ \ : X\,\Gamma\,A \to (\beta : \mathsf{Vars}\,\Psi\,\Gamma) \to X\,\Psi\,A[\ulcorner\beta\urcorner] && \ulcorner[]\urcorner : \ulcorner n\urcorner[\ulcorner\beta\urcorner] \equiv \ulcorner n[\beta]\urcorner\\
&{-}\circ{-} : X\mathsf{s}\,\Gamma\,\Delta \to \mathsf{Vars}\,\Psi\,\Gamma \to X\mathsf{s}\,\Psi\,\Delta && \ulcorner\tau\urcorner \circ \ulcorner\beta\urcorner \equiv \ulcorner\tau\circ\beta\urcorner
\end{aligned}
$$

Now we can define the presheaf $X_\Gamma$ and families of presheaves $X_A$ for any $A : \mathsf{Ty}\,\Gamma$ where X is either NE or NF. The definitions follow that of TM.

$$
\begin{aligned}
&\Gamma : \mathsf{Con} && X_\Gamma\ : \mathsf{PSh}\,\mathsf{REN}^{\mathsf{op}} && X_\Gamma\,\Psi && := X\mathsf{s}\,\Psi\,\Gamma && X_\Gamma\,\beta\,\tau := \tau\circ\beta\\
&A : \mathsf{Ty}\,\Gamma && X_A : \mathsf{FamPSh}\,\mathsf{TM}_\Gamma && X_A\,(\rho : \mathsf{TM}_\Gamma\,\Psi) && := X\,\Psi\,A[\rho] && X_A\,\beta\,n := n[\beta]
\end{aligned}
$$

We set the parameters of the logical predicate at the base type and family by defining $\bar{\mathsf{U}}$ and $\bar{\mathsf{El}}$. The predicate holds for a term if there is a neutral term of the corresponing type which is equal to the term. The action on morphisms is just renaming.

$$
\begin{aligned}
&\bar{\mathsf{U}}\ : \mathsf{FamPSh}\,\mathsf{TM}^{\mathsf{U}} && \bar{\mathsf{U}}_\Psi\,(\hat{A} : \mathsf{Tm}\,\Psi\,\mathsf{U}) && := \Sigma(n : \mathsf{Ne}\,\Psi\,\mathsf{U}).\hat{A} \equiv \ulcorner n\urcorner\\
&\bar{\mathsf{El}} : \mathsf{FamPSh}\left(\Sigma\left(\Sigma\,(\mathsf{TM}^{\mathsf{U}}\,\mathsf{TM}^{\mathsf{El}})\right)\bar{\mathsf{U}}[\mathsf{wk}]\right) && \bar{\mathsf{El}}_\Psi\,(\hat{A}, t : \mathsf{Tm}\,\Psi\,(\mathsf{El}\,\hat{A}), p) && := \Sigma(n : \mathsf{Ne}\,\Psi\,(\mathsf{El}\,\hat{A})).t \equiv \ulcorner n\urcorner
\end{aligned}
$$

Now we can interpret any term in the syntax in the logical predicate model over $\mathsf{REN}^{\mathsf{op}}$ with base type interpretations U and El.

## 7 Quote and unquote

By the logical predicate interpretation using $\bar{\mathsf{U}}$ and $\bar{\mathsf{El}}$ we know the following two things:

- terms at the base types are equal to a normal form,
- this property is preserved by the other type formers (functions and subsituted types).

We make use of this fact to lift the first property to any type. In order to do this, we define a quote function by induction on the type which takes a term which preserves the predicate to a normal form and a proof that it is equal to it. Because of the function space, we need to define a function in the other direction as well, from neutral terms to the predicate.

In fact, we define the quote function q and unquote u by induction on the structure of contexts and types. For this, we need to define a model of type theory for which only the motives for contexts and types are interesting.

The motive for a context $\Delta$ is the following two functions $\mathsf{q}_\Delta$ and $\mathsf{u}_\Delta$ together with their naturality properties. Both have dependent type to express the commutativity of the diagram in 2. Quote takes a substitution for which the predicate holds and returns a normal substitution together with a proof of convertibility. Unquote takes a neutral substitution (list of neutral terms) and returns a proof that the logical predicate holds for this substitution.

Note that in the statement of the naturality conditions we implicitly assume an embedding of renamings into neutral and normal substitutions.

$$
\begin{aligned}
\Delta : \mathsf{Con} \qquad & \Sigma\big(\mathsf{q}_\Delta : \{\Psi : |\mathsf{REN}^{\mathsf{op}}|\}(\rho : \mathsf{TM}_\Delta\,\Psi) \to [\![\Delta]\!]_\Psi\,\rho \to \Sigma(\tau : \mathsf{NF}_\Delta\,\Psi).\rho \equiv \ulcorner\tau\urcorner\big) \\
& \quad .\forall(\beta : \mathsf{Vars}\,\Omega\,\Psi), \rho, \alpha.\mathsf{let}\,(\tau, p) := \mathsf{q}_{\Delta\,\Psi}\,\rho\,\alpha \\
& \qquad\qquad\qquad\qquad \mathsf{in}\;\; (\tau \circ \beta, \mathsf{ap}\,(- \circ \beta)\,p) \equiv \mathsf{q}_{\Delta\,\Omega}\,(\rho \circ \beta, [\![\Delta]\!]\,\beta\,\alpha) \\
& \Sigma\Big(\mathsf{u}_\Delta\;\; : \{\Psi : |\mathsf{REN}^{\mathsf{op}}|\}(\tau : \mathsf{NE}_\Delta\,\Psi) \to [\![\Delta]\!]_\Psi\,\ulcorner\tau\urcorner\Big) \\
& \quad .\forall(\beta : \mathsf{Vars}\,\Omega\,\Psi), n.[\![\Delta]\!]\,\beta\,(\mathsf{u}_{\Delta\,\Psi}\,\tau) \equiv \mathsf{u}_{\Delta\,\Omega}\,(\tau \circ \beta)
\end{aligned}
$$

The motive for a type $A$ is again a quote and unquote function. Because of having dependent types, they depend on a substitution for which the predicate holds, and then map semantic elements into normal terms and neutral terms into semantic elements similarly to quote and unquote for contexts. The naturality conditions again show that they are stable under renamings.

$$
\begin{aligned}
A : \mathsf{Ty}\,\Gamma \qquad & \Sigma\big(\mathsf{q}_A : \{\Psi : |\mathsf{REN}^{\mathsf{op}}|\}(\rho : \mathsf{TM}_\Gamma\,\Psi)(\alpha : [\![\Gamma]\!]\,\rho)(s : \mathsf{TM}_A\,\rho)(a : [\![A]\!]\,(\rho, s, \alpha)) \\
& \qquad\quad \to \Sigma(n : \mathsf{NF}_A\,\rho.s \equiv \ulcorner n\urcorner)\big) \\
& \quad .\forall(\beta : \mathsf{Vars}\,\Omega\,\Psi), \rho, \alpha, s, a \\
& \qquad\qquad .\mathsf{let}\,(n, p) := \mathsf{q}_{A\,\Psi}\,\rho\,\alpha\,s\,a \\
& \qquad\qquad\;\; \mathsf{in}\;\; (n[\beta], \mathsf{ap}\,(-[\beta])\,p) \equiv \mathsf{q}_{A\,\Omega}\,(\rho \circ \beta)\,([\![\Gamma]\!]\,\beta\,\alpha)\,(s[\beta])\,([\![A]\!]\,\beta\,a) \\
& \Sigma\big(\mathsf{u}_A : \{\Psi : |\mathsf{REN}^{\mathsf{op}}|\}(\rho : \mathsf{TM}_\Gamma\,\Psi)(\alpha : [\![\Gamma]\!]\,\rho)(n : \mathsf{NE}_A\,\rho) \to [\![A]\!]\,(\rho, \ulcorner n\urcorner, \alpha)\big) \\
& \quad .\forall(\beta : \mathsf{Vars}\,\Omega\,\Psi), \rho, \alpha, n.[\![A]\!]\,\beta\,(\mathsf{u}_{A\,\Psi}\,\rho\,\alpha\,n) \equiv \mathsf{u}_{A\,\Omega}\,(\rho \circ \beta)\,([\![\Gamma]\!]\,\beta\,\alpha)\,(n[\beta])
\end{aligned}
$$

The motives for substitutions and terms are the constant unit families.

We will list the methods for contexts and types excluding the naturality proofs for brevity. We also checked that the equality methods for types hold, but we also don't list the proofs. The methods for the equality constructors of substitutions and terms are trivial.

Quote and unquote for the empty context is trivial, for extended contexts it is pointwise. $,\equiv$ is the congruence law of substitution extension $-,-$ (the equality constructor for extended substitutions).

$$
\begin{aligned}
\mathsf{q}_\cdot\,\rho\,\alpha \quad & := (\epsilon, \epsilon\eta) \\
\mathsf{u}_\cdot\,\tau \quad & := \mathsf{tt} \\
\mathsf{q}_{\Delta,A}\,\rho\,\alpha \quad & := \mathsf{let}\,(\tau, p)\;\; := \mathsf{q}_\Delta\,(\pi_1\,\rho)\,(\mathsf{proj}_1\,\alpha) \\
& \qquad\quad (n, p') := \mathsf{q}_A\,(\pi_1\,\rho)\,(\mathsf{proj}_1\,\alpha)\,(\pi_2\,\rho)\,(\mathsf{proj}_2\,\alpha) \\
& \qquad \mathsf{in}\;\; \big((\tau, n), (,\equiv\; p\,p')\big) \\
\mathsf{u}_{\Delta,A}\,(\tau, n) & := \mathsf{let}\,\alpha := \mathsf{u}_\Delta\,\tau\;\mathsf{in}\,(\alpha, \mathsf{u}_A\,\ulcorner\tau\urcorner\,\alpha\,n)
\end{aligned}
$$

Quoting or unquoting a substituted type is the same as quoting at the type and using the fundamental theorem at the substitution to lift the witness of the predicate $\alpha$. As expected, quoting at base types is simply returning the witness of the predicate, while unquoting just returns the neutral term itself and the witness of the predicate will be reflexivity.

$$
\begin{aligned}
\mathsf{q}_{A[\sigma]}\,\rho\,\alpha\,s\,a &:= \mathsf{q}_A\,(\sigma \circ \rho)\,([\![\sigma]\!]\,(\rho, \alpha))\,s\,a & \mathsf{u}_{A[\sigma]}\,\rho\,\alpha\,n &:= \mathsf{u}_A\,(\sigma \circ \rho)\,([\![\sigma]\!]\,(\rho, \alpha))\,n \\
\mathsf{q}_{\mathsf{U}}\,\rho\,\alpha\,s\,a &:= a & \mathsf{u}_{\mathsf{U}}\,\rho\,\alpha\,n &:= (n, \mathsf{refl}) \\
\mathsf{q}_{\mathsf{El}\,\hat{A}}\,\rho\,\alpha\,s\,a &:= a & \mathsf{u}_{\mathsf{El}\,\hat{A}}\,\rho\,\alpha\,n &:= (n, \mathsf{refl})
\end{aligned}
$$

The normal form of a function $s$ is $lam\, n$ for some normal form $n$ which is in the extended context. We get this $n$ by quoting $\mathsf{app}\, s$ in the extended context. $f$ is the witness that $s$ preserves the relation for any renaming, and we the renaming $\mathsf{wk}\,\mathsf{id}$ to put $f$ in the extended context. The argument of $f$ in this case will be the zero de Bruijn index $\mathsf{vze}$ and we need to unquote it to get the witness that it preserves the logical predicate. This is the place where the Kripke property of the logical relation is needed: the base category of the Kripke logical relation needs to minimally include the morphism $\mathsf{wk}\,\mathsf{id}$.

$$
\begin{aligned}
\mathsf{q}_{\Gamma\vdash\Pi\, A\, B\, \Psi}\, \rho\, \alpha\, s\, f := \mathsf{let}\; & a \quad\;\, := \mathsf{u}_{\Psi, A[\rho]}\, (\rho \circ \pi_1\, \mathsf{id})\, \big(\llbracket\Gamma\rrbracket\, (\mathsf{wk}\,\mathsf{id})\, \alpha\big)\, \mathsf{vze} \\
& (n, p) := \mathsf{q}_{B\, \Psi, A[\rho]}\, \rho^A\, \big(\llbracket\Gamma\rrbracket\, (\mathsf{wk}\,\mathsf{id})\, \alpha, a\big)\, (\mathsf{app}\, s)\, \big(\mathsf{map}\, f\, (\mathsf{wk}\,\mathsf{id})\, \ulcorner\mathsf{vze}\urcorner\, a\big) \\
\mathsf{in}\; & (\mathsf{lam}\, n, \Pi\eta \centerdot \mathsf{ap}\,\mathsf{lam}\, p)
\end{aligned}
$$

We only show the mapping part of unquoting a function. To show that $n$ preserves the predicate, we show that it preserves the predicate for every argument $u$ for which the predicate holds (by $v$). We quote the arguments, thereby getting it in normal form ($m$), and now we can unquote the neutral term $(app\, n[\beta]\, m)$ to get the result. We also need to transport the result along the proof $p$ that $u \equiv \ulcorner m\urcorner$.

$$
\begin{aligned}
\mathsf{map}\, (\mathsf{u}_{\Gamma\vdash\Pi\, A\, B\, \Psi}\, \rho\, \alpha\, n) := \lambda(\beta : \mathsf{Vars}\, \Omega\, \Psi), u, v\; & \\
.\mathsf{let}\, (m, p) := \mathsf{q}_{A\, \Omega}\, (\rho \circ \beta)\, (\llbracket\Gamma\rrbracket\, \beta\, \alpha)\, u\, v & \\
\mathsf{in}\;\; {}_{p*}\big(\mathsf{u}_{B\, \Omega}\, (\rho \circ \beta, u)\, (\llbracket\Gamma\rrbracket\, \beta\, \alpha, v)\, (\mathsf{app}\, n[\beta]\, m)\big) &
\end{aligned}
$$

## 8 Normalisation

Now we can define the normalisation function and show that it is complete as follows.

$$
\begin{aligned}
\mathsf{norm}_A\, (t : \mathsf{Tm}\, \Gamma\, A) \;\; &: \mathsf{Nf}\, \Gamma\, A && := \mathsf{proj}_1\, \big(\mathsf{q}_A\, \mathsf{id}_\Gamma\, (\mathsf{U}_\Gamma\, \mathsf{id}_\Gamma)\, t\, (\llbracket t\rrbracket\, \mathsf{id}_\Gamma\, (\mathsf{U}_\Gamma\, \mathsf{id}_\Gamma))\big) \\
\mathsf{compl}_A\, (t : \mathsf{Tm}\, \Gamma\, A) &: t \equiv \ulcorner\mathsf{norm}_A\, t\urcorner && := \mathsf{proj}_2\, \big(\mathsf{q}_A\, \mathsf{id}_\Gamma\, (\mathsf{U}_\Gamma\, \mathsf{id}_\Gamma)\, t\, (\llbracket t\rrbracket\, \mathsf{id}_\Gamma\, (\mathsf{U}_\Gamma\, \mathsf{id}_\Gamma))\big)
\end{aligned}
$$

We prove stability by mutual induction on normal forms and neutral terms.

$$
\frac{n : \mathsf{Nf}\, \Gamma\, A}{\mathsf{norm}_A\, \ulcorner n\urcorner \equiv n}
\qquad
\frac{n : \mathsf{Ne}\, \Gamma\, A}{\llbracket\ulcorner n\urcorner\rrbracket\, (\mathsf{id}_\Gamma, \mathsf{U}_\Gamma\, \mathsf{id}_\Gamma) \equiv \mathsf{u}_A\, \mathsf{id}_\Gamma\, (\mathsf{U}_\Gamma\, \mathsf{id}_\Gamma)\, n}
$$

Decidability of normal forms is proven by mutual induction on normal forms, neutral terms and types. Because of type dependencies, we decide equality in the total space $\Sigma(\Gamma : \mathsf{Con}).\Sigma(A : \mathsf{Ty}\, \Gamma).\mathsf{Nf}\, \Gamma\, A$. When deciding whether two applications are equal, we need to first compare the types of domain and codomain of the function, and we only proceed with comparing the neutral functions when we know that they are equal. To decide equality of types, we need to normalise them and this includes normalisation of terms of type $\mathsf{U}$. $\mathsf{isDec}\, A$ is defined by $A + (A \to \bot)$.

$$
\frac{n, n' : \mathsf{Nf}\, \Gamma\, A}{\mathsf{isDec}\, \big((\Gamma, A, n) \equiv (\Gamma, A, n')\big)}
\qquad
\frac{A \equiv A' \quad n : \mathsf{Nf}\, \Gamma\, A \quad n' : \mathsf{Nf}\, \Gamma\, A'}{\mathsf{isDec}\, \big((\Gamma, A, n) \equiv (\Gamma, A', n')\big)}
\qquad
\frac{A, A' : \mathsf{Ty}\, \Gamma}{\mathsf{isDec}\, \big((\Gamma, A) \equiv (\Gamma, A')\big)}
$$

## 9 Conclusions and further work

We proved normalisation for a basic dependent type theory with the technique of NBE. To our knowledge, this is the first presentation of NBE using typed terms (also called intrinsic typing). The advantage over approaches using realizers is its simplicity. Compared to previous categorical normalisation proofs such as [6] we need fewer constructions as we replace the presheaf model and a logical relation by a logical predicat (which is however proof relevant). In our approach the normalisation function and the proof of correctness are defined mutually, there seems to be no way of separating them. Another property to note is that we don't normalise types, we just index terms by not necessarily normal types.

We are currently working on completing the formalisation[5] [9]. Most of the work here is equality reasoning. QIITs make it possible to define the syntax of type theory in a very concise way, however because of the lacking of computation rules, using them involves large amounts of boilerplate. We expect that a cubical metatheory [11], with its organised way of expressing equalities depending on equalities and its additional computation rules would reduce the amount of boilerplate significantly.

Another challange is to extend our basic type theory by $\Sigma$ types, inductive types, universes and large elimination. Also, it would be interesting to see how the work fits into the setting of homotopy type theory (without assuming $\mathsf{K}$). Another question to investigate is whether the logical predicate interpretation can be generalised to work over arbitrary presheaf models and how the syntactic model fits here.

## References


1. The Agda Wiki, 2015. Available online.
2. Andreas Abel. Towards normalization by evaluation for the $\beta\eta$-calculus of constructions. In *Functional and Logic Programming*, pages 224–239. Springer, 2010.
3. Andreas Abel, Thierry Coquand, and Peter Dybjer. Normalization by evaluation for Martin-Löf type theory with typed equality judgements. In *Logic in Computer Science, 2007. LICS 2007. 22nd Annual IEEE Symposium on*, pages 3–12. IEEE, 2007.
4. Andreas Abel and Habilitation Talk. *Normalization by Evaluation: Dependent Types and Impredicativity*. PhD thesis, Habilitation, Ludwig-Maximilians-Universität München, 2013.
5. Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Phil Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 303–310, 2001.
6. Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Categorical reconstruction of a reduction free normalization proof. In David Pitt, David E. Rydeheard, and Peter Johnstone, editors, *Category Theory and Computer Science*, LNCS 953, pages 182–199, 1995.
7. Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Reduction-free normalisation for a polymorphic system. In *11th Annual IEEE Symposium on Logic in Computer Science*, pages 98–106, 1996.
8. Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Reduction-free normalisation for system $F$. available from `http://www.cs.nott.ac.uk/~txa/publ/f97.pdf`, 1997.
9. Thorsten Altenkirch and Ambrus Kaposi. Supplementary material for the paper Normalisation by Evaluation for Dependent Types, 2016. Available online at the second author's website.
10. Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL 2016, pages 18–29, New York, NY, USA, 2016. ACM.


[5] The current status of formalisation is that we formalised all the main constructions but the functoriality and naturality properties are left as holes.

11 Simon Huber Cyril Cohen, Thierry Coquand and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. December 2015.

12 Nils Anders Danielsson. A formalisation of a dependently typed language as an inductive-recursive family. In *Types for Proofs and Programs*, pages 93–109. Springer, 2006.

13 Peter Dybjer. Internal type theory. In *Types for Proofs and Programs*, pages 120–134. Springer, 1996.

14 Martin Hofmann. Syntax and semantics of dependent types. In *Extensional Constructs in Intensional Type Theory*, pages 13–54. Springer, 1997.

15 Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007.

16 Steven Schäfer, Gert Smolka, and Tobias Tebbi. Completeness and decidability of de bruijn substitution algebra in coq. In *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 67–73. ACM, 2015.