Normalisation by Evaluation for Dependent Types*

Thorsten Altenkirch[1] and Ambrus Kaposi[2]

1 **School for Computer Science, University of Nottingham**
Nottingham, United Kingdom
txa@cs.nott.ac.uk

2 **Department of Programming Languages and Compilers,**
Eötvös Loránd University, Budapest, Hungary
akaposi@inf.elte.hu

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.FSCD.2016.n

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 Γ 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 lam and app, it also has equality constructors e.g. expressing the β 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 norm which takes a term to a normal form. It needs to be an isomorphism:

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

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

* Supported by EPSRC grant EP/M016951/1 and USAF grant FA9550-16-1-0029.

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 every function defined from the syntax respects the equality constructors.

1.2 NBE for simple type theory

Normalisation by evaluation (NBE) is one way to implement this specification. In this subsection, we summarize the approach of [6]. NBE works by evaluating the syntax in a presheaf model over the category of renamings REN and with normal forms as interpretation of the base type. The objects in REN are contexts and morphisms are lists of variables. Note that for any context Γ 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.

$$\mathsf{TM}_\Delta : \mathsf{PSh}\,\mathsf{REN} \qquad \mathsf{NE}_\Delta : \mathsf{PSh}\,\mathsf{REN} \qquad \mathsf{NF}_\Delta : \mathsf{PSh}\,\mathsf{REN}$$
$$\mathsf{TM}_\Delta\,\Gamma := \mathsf{Tms}\,\Gamma\,\Delta \qquad \mathsf{NE}_\Delta\,\Gamma := \mathsf{Nes}\,\Gamma\,\Delta \qquad \mathsf{NF}_\Delta\,\Gamma := \mathsf{Nfs}\,\Gamma\,\Delta$$

To normalise a substitution with codomain Δ, one defines two natural transformations unquote u_Δ and quote q_Δ by induction on the structure of contexts and types such that the diagram in figure 1 commutes. $[\![\Delta]\!]$ denotes the interpretation of Δ in the presheaf model and R_Δ denotes the logical relation at context Δ between TM_Δ and $[\![\Delta]\!]$. The logical relation is equality at the base type.

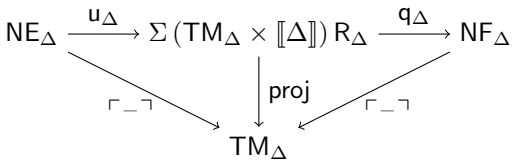

Figure 1 The type of quote and unquote for a context Δ in NBE for simple types.

Now a substitution σ 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 R_σ which also needs two related elements: these are given by unquoting the identity renaming (which is neutral).

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

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.

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

In the case of (dependent) type theory, types depend on contexts, so $\mathsf{TM}_{\Gamma\vdash A}$ becomes a family of presheaves over TM_Γ, $[\![\Gamma \vdash A]\!]$ is a family over $[\![\Gamma]\!]$ and $\mathsf{R}_{\Gamma\vdash A}$ depends on R_Γ (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 conditions 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 Π 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 Δ by $[\![\Delta]\!]$. We define normalisation following the diagram in figure 2.

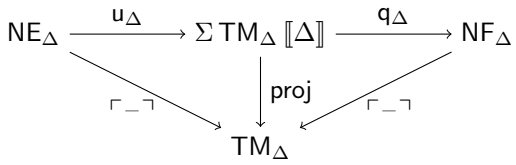

Figure 2 The type of quote and unquote for a context Δ in our proof.

In the presheaf model, the interpretation of the 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 exists a normal form which is equal to the term.

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 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 REN as the base category and two parameters for the interpretations of U and El. This interpretation can be seen as a dependent version of the presheaf model of type theory. E.g. a context in the presheaf model is interpreted as a

presheaf. Now it is a family of presheaves dependent on a substitution into that context. The interpretations of base types can depend on the actual elements of the base types. The interpretation of substitutions and terms are the 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 U and El. The interpretation of U at a term of type U will be a neutral term of type 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 and completeness are given by interpreting the term in the logical predicate model at the identity semantic element and then quoting. Stability is proved by mutual induction on neutral terms and normal forms.

1.5 Related work

Normalisation by evaluation was first formulated by Schwichtenberg and Berger [11], subsequently a categorical account using presheaf categories was given [6] and this approach was extended to System F [7, 8] and coproducts [5]. The present work can be seen as a continuation of this line of research.

The term normalisation by evaluation is also more generally used to describe semantic based normalisation functions. E.g. Danvy is using semantic normalisation for partial evaluation [14]. Normalisation by evaluation using untyped realizers has been applied to dependent types by Abel et al [2–4]. Danielsson [13] has formalized NBE for dependent types but he doesn't prove soundness of normalisation.

2 Metatheory and notation

We are working in intensional Martin-Löf Type Theory with postulated extensionality principles using Agda as a vehicle [1, 17]. We extend Agda with quotient inductive inductive types (QIITs, see section 6 of [10]) using axioms. When defining an inductive type A, we first 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 K, that is, we work in a strict type theory.

We follow Agda's convention of denoting the universe of types by Set, we write function types as $(x : A) \to B$ or $\forall x.B$, implicit arguments are written in curly braces $\{x : A\} \to B$ and can be omitted or given in lower index. If some arguments are omitted, we assume universal quantification, e.g. $(y : B\, x) \to C$ means $\forall\, x.(y : B\, x) \to C$ if x is not given in the context. We write $\Sigma(x : A).B$ for Σ types. We overload names e.g. 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 refl. Transport of a term $a : P\, x$ along an equality $p : x \equiv y$ is denoted ${}_{p*}a : P\, y$. We denote $({}_{p*}a) \equiv b$ by $a \equiv^p b$. We write 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, e.g. we

might write $f\,a$ where $f : A \to B$ and $a : A'$ but in this case there is an equality in scope which justifies $A \equiv A'$.

Sometimes we use Coq-style definitions: we write $\mathsf{d}\,(x : A) : B := t$ for defining d of 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.

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

We use the convention of naming contexts Γ, Δ, Θ, types A, B, terms t, u, substitutions σ, ν, δ.

We define a basic type theory with an uninterpreted base type U, a family over this type El and dependent function space Π with constructor lam and eliminator app. Our type theory is given as an explicit substitution calculus, hence the QIIT needs constructors $-[-]$ for substituted types and terms. The constructors of the QIIT 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 [18] 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 π_1, π_2.

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

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{data}} \\
\cdot & : \mathsf{Con} \\
-,- & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
-[-] & : \mathsf{Ty}\,\Delta \to \mathsf{Tms}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma \\
\mathsf{U} & : \mathsf{Ty}\,\Gamma \\
\mathsf{El} & : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma \\
\Pi & : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma, A) \to \mathsf{Ty}\,\Gamma \\
\mathsf{id} & : \mathsf{Tms}\,\Gamma\,\Gamma \\
- \circ - & : \mathsf{Tms}\,\Theta\,\Delta \to \mathsf{Tms}\,\Gamma\,\Theta \to \mathsf{Tms}\,\Gamma\,\Delta \\
\epsilon & : \mathsf{Tms}\,\Gamma\,\cdot \\
-,- & : (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \to \mathsf{Tms}\,\Gamma\,(\Delta, A) \\
\pi_1 & : \mathsf{Tms}\,\Gamma\,(\Delta, A) \to \mathsf{Tms}\,\Gamma\,\Delta \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Tms}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \\
\pi_2 & : (\sigma : \mathsf{Tms}\,\Gamma\,(\Delta, A)) \to \mathsf{Tm}\,\Gamma\,A[\pi_1\,\sigma] \\
\mathsf{lam} & : \mathsf{Tm}\,(\Gamma, A)\,B \to \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \\
\mathsf{app} & : \mathsf{Tm}\,\Gamma\,(\Pi\,A\,B) \to \mathsf{Tm}\,(\Gamma, A)\,B
\end{array}
\qquad
\begin{array}{ll}
\multicolumn{2}{l}{\mathsf{data}} \\
[\mathsf{id}] & : A[\mathsf{id}] \equiv A \\
[][] & : A[\sigma][\nu] \equiv A[\sigma \circ \nu] \\
\mathsf{U}[] & : \mathsf{U}[\sigma] \equiv \mathsf{U} \\
\mathsf{El}[] & : (\mathsf{El}\,\hat{A})[\sigma] \equiv \mathsf{El}\,({}_{\mathsf{U}[]*}\hat{A}[\sigma]) \\
\Pi[] & : (\Pi\,A\,B)[\sigma] \equiv \Pi\,(A[\sigma])\,(B[\sigma^A]) \\
\mathsf{id}\circ & : \mathsf{id} \circ \sigma \equiv \sigma \\
\circ\mathsf{id} & : \sigma \circ \mathsf{id} \equiv \sigma \\
\circ\circ & : (\sigma \circ \nu) \circ \delta \equiv \sigma \circ (\nu \circ \delta) \\
\epsilon\eta & : \{\sigma : \mathsf{Tms}\,\Gamma\,\cdot\} \to \sigma \equiv \epsilon \\
\pi_1\beta & : \pi_1\,(\sigma, t) \equiv \sigma \\
\pi\eta & : (\pi_1\,\sigma, \pi_2\,\sigma) \equiv \sigma \\
,\circ & : (\sigma, t) \circ \nu \equiv (\sigma \circ \nu), ({}_{[][]*}t[\nu]) \\
\pi_2\beta & : \pi_2\,(\sigma, t) \equiv^{\pi_1\beta} t \\
\Pi\beta & : \mathsf{app}\,(\mathsf{lam}\,t) \equiv t \\
\Pi\eta & : \mathsf{lam}\,(\mathsf{app}\,t) \equiv t \\
\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 transport 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 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 the 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 Dybjer's Categories with Families [15].

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

[3] Dybjer uses the usual application operator, we use the categorical one, the projections π_1, π_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.

names of the constructors followed by an upper index 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 equality constructor $\pi_2\beta$.

4 The category of renamings

In this section we define the category of renamings REN. Objects in this category are contexts, morphisms are renamings (Vars): lists of de Bruijn variables.

We define the types of variables Var and renamings 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.

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

Variables are typed de Bruijn indices. vze projects out the last element of the context, vsu extends the context, and the type $A : \mathsf{Ty}\,\Psi$ needs to be weakened in both cases because we need to interpret it in Ψ 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 names Ψ, Ω, Ξ for objects of REN, x, y for variables, β, γ 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.

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

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 [16].

A contravariant presheaf over a category $\mathcal{C}$ is denoted $\Gamma : \mathsf{PSh}\,\mathcal{C}$. 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 Γ is denoted $A : \mathsf{FamPSh}\,\Gamma$. It is given by the following data [4] : 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 Γ and Δ 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[5] from a presheaf Γ to a family of presheaves A over Γ is denoted $t : \Gamma \overset{\mathsf{s}}{\to} A$. 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 $\Gamma : \mathsf{PSh}\,\mathcal{C}$ and $A : \mathsf{FamPSh}\,\Gamma$ we can define $\Sigma\,\Gamma\,A : \mathsf{PSh}\,\mathcal{C}$ 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 $A[\sigma]\,f\,a := {}_{\mathsf{natn}\,\sigma *}(A\,f\,a)$ for $\alpha : \Gamma\,I$, $a : A[\sigma]\,\alpha$ 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 $t : \Gamma \overset{\mathsf{s}}{\to} A$ 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 by $[\![t]\!]$. The following table gives the motives of the eliminator.

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

[4] Indeed, this is equivalent to a presheaf over the category of elements $\int \Gamma$.

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

First we define the syntactic presheaf interpretation TM as given in the table. TM_Δ is a presheaf over REN, the action on morphisms is $\mathsf{TM}_\Delta\,(\beta : \mathsf{Vars}\,\Omega\,\Psi)\,\sigma := \sigma \circ \ulcorner\beta\urcorner$. TM_A is a family of presheaves over TM_Γ, TM_σ is a natural transformation and TM_t is a section. The action on morphisms and the functor laws for TM_A and the naturality laws for TM_σ and TM_t are straightforward. 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.

In the logical predicate interpretation, a context Δ is mapped to a family of presheaves over TM_Δ. 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 ρ. Moreover, we have the renaming $[\![\Delta]\!]\,\beta : [\![\Delta]\!]\,\rho \to [\![\Delta]\!]\,(\mathsf{TM}_\Delta\,\beta\,\rho)$.

$[\![A]\!]$ 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.

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

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

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

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

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

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

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

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

We define the presheaf $\mathsf{TM}^{\mathsf{U}} : \mathsf{PSh}\,\mathsf{REN}$ 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 action on a morphism β is just substitution $-[\ulcorner\beta\urcorner]$ for both.

Note that the base category of the logical predicate interpretation is fixed to REN. However we parameterise the interpretation by the predicate at the base type U and base family 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 ρ 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 $\_*\_$ e.g. $[\![\Gamma]\!]\,\beta\,\alpha$ 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.

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

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 β in a natural way. Renaming a witness of the logical predicate at the function type is postcomposing the Kripke morphism by it.

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

Now we list the methods for the substitution constructors, that is, we prove the fundamental theorem for substitutions. We omit the naturality proofs. The 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 theorem for lam and app are more interesting. For lam, the 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 ρ and the witness of the predicate α to account for the Kripke property. The naturality is given by the naturality of the term itself. Application uses the map part of the logical predicate and the identity renaming.

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

Lastly, we need to provide methods for the equality constructors. We won't list all of these proofs as they are quite straightforward, but as examples we show the semantic versions of the laws $[][]$ and $\pi_2\beta$. For $[][]$, we have to show that the two families of presheaves $\llbracket A[\sigma][\nu]\rrbracket$ and $\llbracket A[\sigma\circ\nu]\rrbracket$ are equal. It is enough to show that their action on objects and morphisms coincides as the equalities will be equal by K. Note that we use function extensionality to show the equality of the presheaves from the pointwise equality of actions. When we unfold the definitions for the actions on objects we see that the results are equal by associativity.

$$\begin{aligned}&\llbracket A[\sigma][\nu]\rrbracket\,(\rho,s,\alpha)\\ =\;&\llbracket A\rrbracket\,\Big(\sigma\circ(\nu\circ\rho),s,\llbracket\sigma\rrbracket\,\big(\nu\circ\rho,\llbracket\nu\rrbracket\,(\rho,\alpha)\big)\Big)\\ \equiv\;&\llbracket A\rrbracket\,\Big((\sigma\circ\nu)\circ\rho,s,\llbracket\sigma\rrbracket\,\big(\nu\circ\rho,\llbracket\nu\rrbracket\,(\rho,\alpha)\big)\Big)\\ =\;&\llbracket A[\sigma\circ\nu]\rrbracket\,(\rho,s,\alpha)\end{aligned}$$

The actions on morphisms are equal by unfolding the definitions.

$$\llbracket A[\sigma][\nu]\rrbracket\,\beta\,a = \llbracket A\rrbracket\,\beta\,a = \llbracket A[\sigma\circ\nu]\rrbracket\,\beta\,a$$

For $\pi_2\beta$ we need to show that two sections $\llbracket\pi_2\,(\sigma,t)\rrbracket$ and $\llbracket t\rrbracket$ are equal, and again, the law parts of the sections will be equal by K.

$$\llbracket\pi_2\,(\sigma,t)\rrbracket\,(\rho,\alpha) = \mathsf{proj}_2\,\big(\llbracket\sigma,t\rrbracket\,(\rho,\alpha)\big) = \llbracket t\rrbracket\,(\rho,\alpha)$$

6 Normal forms

We define η-long β-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.

$$\begin{aligned}&\mathsf{data}\;\mathsf{Ne} : (\Gamma:\mathsf{Con})\to\mathsf{Ty}\,\Gamma\to\mathsf{Set}\\ &\mathsf{data}\;\mathsf{Nf} : (\Gamma:\mathsf{Con})\to\mathsf{Ty}\,\Gamma\to\mathsf{Set}\\ &\ulcorner\_\urcorner \quad : \mathsf{Nf}\,\Gamma\,A\to\mathsf{Tm}\,\Gamma\,A\\ &\mathsf{data}\;\mathsf{Ne}\\ &\quad \mathsf{var} \quad : \mathsf{Var}\,\Gamma\,A\to\mathsf{Ne}\,\Gamma\,A\\ &\quad \mathsf{app} \quad : \mathsf{Ne}\,\Gamma\,(\Pi\,A\,B)\to(v:\mathsf{Nf}\,\Gamma\,A)\\ &\qquad\quad \to\mathsf{Ne}\,\Gamma\,(B[<\ulcorner v\urcorner>])\end{aligned}$$

$$\begin{aligned}&\mathsf{data}\;\mathsf{Nf}\\ &\quad \mathsf{neuU} \;\; : \mathsf{Ne}\,\Gamma\,\mathsf{U}\to\mathsf{Nf}\,\Gamma\,\mathsf{U}\\ &\quad \mathsf{neuEl} : \mathsf{Ne}\,\Gamma\,(\mathsf{El}\,\hat{A})\to\mathsf{Nf}\,\Gamma\,(\mathsf{El}\,\hat{A})\\ &\quad \mathsf{lam} \;\; : \mathsf{Nf}\,(\Gamma,A)\,B\to\mathsf{Nf}\,\Gamma\,(\Pi\,A\,B)\\ &\ulcorner\_\urcorner \quad : \mathsf{Ne}\,\Gamma\,A\to\mathsf{Tm}\,\Gamma\,A\end{aligned}$$

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

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

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_Γ 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.

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

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 corresponding type which is equal to the term. The action on morphisms is just renaming.

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

Now we can interpret any term in the logical predicate model over REN with base type interpretations $\bar{\mathsf{U}}$ and $\bar{\mathsf{El}}$. We denote the interpretation of t by $[\![t]\!]$.

7 Quote and unquote

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

- terms at the base type and base family are equal to a normal form,
- this property is preserved by the other type formers — this is what the logical predicate says at function types and substituted types.

We make use of this fact to lift the first property to any type. We do this by defining a quote function by induction on the type. Quote takes a term which preserves the predicate and maps it to a normal form that it is equal to it. Because of function spaces, we need a function in the other direction as well, mapping neutral terms to the witness of the predicate.

More precisely, 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 in which only the motives for contexts and types are interesting.

First we define families of presheaves for contexts and types which express that there is an equal normal form. The actions on objects are given as follows.

$$\mathsf{NF}^{\equiv}{}_{\Delta} : \mathsf{FamPSh}\,\mathsf{TM}_{\Delta} \qquad \mathsf{NF}^{\equiv}{}_{A} : \mathsf{FamPSh}\,(\Sigma\,\mathsf{TM}_{\Gamma}\,\mathsf{TM}_{A})$$
$$\mathsf{NF}^{\equiv}{}_{\Delta}\,(\rho : \mathsf{TM}_{\Delta}\,\Psi) := \Sigma(\rho' : \mathsf{NF}_{\Delta}\,\Psi).\rho \equiv \ulcorner \rho' \urcorner \qquad \mathsf{NF}^{\equiv}{}_{A}\,(\rho, s) := \Sigma(s' : \mathsf{NF}_{A}\,\rho).s \equiv \ulcorner s' \urcorner$$

We use these to write down the motives for contexts and types. We use sections to express the commutativity of the diagram in figure 2. We only write Σ once for iterated usage.

$$\mathsf{u}_{\Delta} : \mathsf{NE}_{\Delta} \xrightarrow{\mathsf{s}} [\![\Delta]\!][\ulcorner - \urcorner] \qquad \mathsf{u}_{A} : \Sigma\,\mathsf{TM}_{\Gamma}\,\mathsf{NE}_{A}\,([\![\Gamma]\!][\mathsf{wk}]) \xrightarrow{\mathsf{s}} [\![A]\!][\mathsf{id}, \ulcorner - \urcorner, \mathsf{id}]$$
$$\mathsf{q}_{\Delta} : \Sigma\,\mathsf{TM}_{\Delta}\,[\![\Delta]\!] \xrightarrow{\mathsf{s}} \mathsf{NF}^{\equiv}{}_{\Delta}[\mathsf{wk}] \qquad \mathsf{q}_{A} : \Sigma\,\mathsf{TM}_{\Gamma}\,\mathsf{TM}_{A}\,([\![\Gamma]\!][\mathsf{wk}])\,[\![A]\!] \xrightarrow{\mathsf{s}} \mathsf{NF}^{\equiv}{}_{A}[\mathsf{wk}][\mathsf{wk}]$$

Unquote for a context takes a neutral substitution and returns a proof that the logical predicate holds for it. Quote takes a substitution for which the predicate holds and returns a normal substitution together with a proof that the original substitution is equal (convertible) to the normal one (embedded into substitutions by $\ulcorner - \urcorner$). The types of unquote and quote for types are more involved as they depend on a substitution for which the predicate needs to hold. Unquote for a type takes such a substitution and a neutral term at the type substituted by this substitution and returns a proof that the predicate holds for this neutral term. The natural transformation $\mathsf{id}, \ulcorner - \urcorner, \mathsf{id}$ is defined in the obvious way, it just embeds the second component (the neutral term) into terms. Quote for a type takes a term of this type for which the predicate holds and returns a normal form at this type together with a proof that it is equal to the term. Here again, another substitution is involved.

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.

Unquote and quote for the empty context are trivial, for extended contexts they are pointwise. ap, is the congruence law of substitution extension $-, -$.

$$\mathsf{u}.\,(\tau : \mathsf{NE}.\,\Psi) : \top := \mathsf{tt}$$
$$\mathsf{q}.\,\big((\sigma : \mathsf{TM}.\,\Psi), (\alpha : \top)\big) : \Sigma(\rho' : \mathsf{NF}.\,\Psi).\rho \equiv \ulcorner \rho' \urcorner := (\epsilon, \epsilon\eta)$$
$$\mathsf{u}_{\Delta,A}\,\big((\tau, n) : \mathsf{NE}_{\Delta,A}\,\Psi\big) : \Sigma\big(\alpha : [\![\Delta]\!]\,(\pi_1 \ulcorner \tau, n \urcorner)\big).[\![A]\!]\,(\pi_1 \ulcorner \tau, n \urcorner, \pi_2 \ulcorner \tau, n \urcorner, \alpha)$$
$$:= \mathsf{u}_{\Delta}\,\tau, \mathsf{u}_{A}\,(\ulcorner \tau \urcorner, n, \mathsf{u}_{\Delta}\,\tau)$$
$$\mathsf{q}_{\Delta,A}\,\big((\rho : \mathsf{TM}_{\Delta}\,\Psi), (\alpha, a) : [\![\Delta, A]\!]\,\rho\big) : \Sigma(\rho' : \mathsf{NF}_{\Delta,A}\,\Psi).\rho \equiv \ulcorner \rho' \urcorner$$
$$:= \mathsf{let}\,(\tau, p) := \mathsf{q}_{\Delta}\,(\pi_1\,\rho, \alpha);\ (n, q) := \mathsf{q}_{A}\,(\pi_1\,\rho, \pi_2\,\rho, \alpha, a)\,\mathsf{in}\,\big((\tau, n), (\mathsf{ap}, p\,q)\big)$$

(Un)quoting a substituted type is the same as (un)quoting at the type and using the fundamental theorem at the substitution to lift the witness of the predicate α. As expected, unquoting at the base type is simply returning the neutral term itself and the witness of the predicate will be reflexivity, while quote just returns the witness of the predicate.

$$\mathsf{u}_{A[\sigma]}\,(\rho, n, \alpha) \quad : [\![A]\!]\,\big(\sigma \circ \rho, \ulcorner n \urcorner, [\![\sigma]\!]\,(\rho, \alpha)\big) := \mathsf{u}_{A}\,\big(\sigma \circ \rho, n, [\![\sigma]\!]\,(\rho, \alpha)\big)$$
$$\mathsf{q}_{A[\sigma]}\,(\rho, s, \alpha, a) : \Sigma(s' : \mathsf{NF}_{A[\sigma]}\,\rho).s \equiv \ulcorner s' \urcorner \quad := \mathsf{q}_{A}\,\big(\sigma \circ \rho, s, [\![\sigma]\!]\,(\rho, \alpha), a\big)$$

$$\mathsf{u}_{\mathsf{U}}\,\big((\rho : \mathsf{TM}_{\Gamma}\,\Psi), (n : \mathsf{Ne}\,\Psi\,\mathsf{U}[\rho]), \alpha\big) : \Sigma(n' : \mathsf{NF}_{\mathsf{U}}\,\mathsf{id}).\ulcorner n \urcorner \equiv \ulcorner n' \urcorner := \mathsf{neuU}\,({}_{\mathsf{U}[]*}n), \mathsf{refl}$$
$$\mathsf{q}_{\mathsf{U}}\,\Big(\rho, t, \alpha, \big(a : \mathsf{NF}^{\equiv}{}_{\mathsf{U}}\,(\mathsf{id}, t)\big)\Big) \quad : \mathsf{NF}^{\equiv}{}_{\mathsf{U}}\,(\rho, t) \quad := {}_{\mathsf{U}[]*}a$$

$$\mathsf{u}_{\mathsf{El}\,\hat{A}}\,\big((\rho : \mathsf{TM}_{\Gamma}\,\Psi), (n : \mathsf{Ne}\,\Psi\,(\mathsf{El}\,\hat{A}[\rho])), \alpha\big) : \Sigma(n' : \mathsf{NF}_{\mathsf{El}\,\hat{A}}\,\mathsf{id}).\ulcorner n \urcorner \equiv \ulcorner n' \urcorner$$
$$:= \mathsf{neuEl}\,({}_{\mathsf{El}[]*}n), \mathsf{refl}$$
$$\mathsf{q}_{\mathsf{El}\,\hat{A}}\,\Big(\rho, t, \alpha, \big(a : \mathsf{NF}^{\equiv}{}_{\mathsf{El}\,\hat{A}}\,(\mathsf{id}, t)\big)\Big) \quad : \mathsf{NF}^{\equiv}{}_{\mathsf{El}\,\hat{A}}\,(\rho, t) \quad := {}_{\mathsf{El}[]*}a$$

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 argument, thereby getting it in normal form (m), and now we can unquote the neutral term ($\mathsf{app}\, n[\beta]\, m$) to get the result. We also need to transport the result along the proof p that $u \equiv \ulcorner m \urcorner$.

$$\begin{aligned}
&\mathsf{map}\Big(\mathsf{u}_{\Pi\, A\, B}\, \big((\rho : \mathsf{TM}_\Gamma\, \Psi), (n : \mathsf{NE}_{\Pi\, A\, B}\, \rho), \alpha\big)\Big)\big(\beta : \mathsf{Vars}\, \Omega\, \Psi\big)\big(u : \mathsf{TM}_A\, (\rho \circ \ulcorner \beta \urcorner)\big) \\
&\quad \big(v : [\![A]\!]_\Omega\, (\rho \circ \ulcorner \beta \urcorner, u, [\![\Gamma]\!]\, \beta\, \alpha)\big)\ :\ [\![B]\!]_\Omega\, \big((\rho \circ \ulcorner \beta \urcorner, u), (\ulcorner n \urcorner[\ulcorner \beta \urcorner])\$u, ([\![\Gamma]\!]\, \beta\, \alpha, v)\big) \\
&:= \mathsf{let}\, (m, p) := \mathsf{q}_A\, (\rho \circ \ulcorner \beta \urcorner, u, [\![\Gamma]\!]\, \beta\, \alpha, v)\ \mathsf{in}\ \mathsf{u}_B\, \big((\rho \circ \ulcorner \beta \urcorner, u), ({}_{p*}\mathsf{app}\, n[\beta]\, m), ([\![\Gamma]\!]\, \beta\, \alpha, v)\big)
\end{aligned}$$

The normal form of a function t is $\mathsf{lam}\, n$ for some normal form n which is in the extended context. We get this n by quoting $\mathsf{app}\, t$ in the extended context. f is the witness that t preserves the relation for any renaming, and we use the renaming $\mathsf{wk}\, \mathsf{id}$ to use f in the extended context. The argument of f in this case will be the zero de Bruijn index 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 $\mathsf{wk}\, \mathsf{id}$.

$$\begin{aligned}
&\mathsf{q}_{\Pi\, A\, B}\, (\rho, t, \alpha, f) : \Sigma(t' : \mathsf{NF}_{\Pi\, A\, B}\, \rho).t \equiv \ulcorner t' \urcorner \\
&\quad := \mathsf{let}\, a \qquad := \mathsf{u}_A\, (\rho \circ \ulcorner \mathsf{wk}\, \mathsf{id} \urcorner, \mathsf{var}\, \mathsf{vze}, [\![\Gamma]\!]\, (\mathsf{wk}\, \mathsf{id})\, \alpha) \\
&\qquad\quad (n, p) := \mathsf{q}_B\, \big(\rho^A, \mathsf{app}\, t, ([\![\Gamma]\!]\, (\mathsf{wk}\, \mathsf{id})\, \alpha, a), \mathsf{map}\, f\, (\mathsf{wk}\, \mathsf{id})\, \ulcorner \mathsf{vze} \urcorner\, a\big) \\
&\qquad \mathsf{in}\ \ (\mathsf{lam}\, n, \Pi\eta^{-1} \centerdot \mathsf{ap}\, \mathsf{lam}\, p)
\end{aligned}$$

We have to verify the equality laws for types. Note that we use function extensionality to show that the corresponding quote and unquote functions are equal. The naturality proofs will be equal by K.

(Un)quote preserves [id] by the left identity law.

$$\begin{aligned}
&\mathsf{u}_{A[\mathsf{id}]}\, (\rho, n, \alpha) \quad = \mathsf{u}_A\, (\mathsf{id} \circ \rho, n, \alpha) \quad \equiv \mathsf{u}_A\, (\rho, n, \alpha) \\
&\mathsf{q}_{A[\mathsf{id}]}\, (\rho, s, \alpha, a) = \mathsf{q}_A\, (\mathsf{id} \circ \rho, s, \alpha, a) \equiv \mathsf{q}_A\, (\rho, s, \alpha, a)
\end{aligned}$$

(Un)quote preserves [][] by associativity for substitutions.

$$\begin{aligned}
&\mathsf{u}_{A[\sigma][\nu]}\, (\rho, n, \alpha) \\
&= \mathsf{u}_A\, \big(\sigma \circ (\nu \circ \rho), n, [\![\sigma]\!]\, (\nu \circ \rho, [\![\nu]\!]\, (\rho, \alpha))\big) \\
&\equiv \mathsf{u}_A\, \big((\sigma \circ \nu) \circ \rho, n, [\![\sigma]\!]\, (\nu \circ \rho, [\![\nu]\!]\, (\rho, \alpha))\big) \\
&= \mathsf{u}_{A[\sigma \circ \nu]}\, (\rho, n, \alpha)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{q}_{A[\sigma][\nu]}\, (\rho, s, \alpha, a) \\
&= \mathsf{q}_A\, \big(\sigma \circ (\nu \circ \rho), s, [\![\sigma]\!]\, (\nu \circ \rho, [\![\nu]\!]\, (\rho, \alpha)), a\big) \\
&\equiv \mathsf{q}_A\, ((\sigma \circ \nu) \circ \rho, s, [\![\sigma]\!]\, (\nu \circ \rho, [\![\nu]\!]\, (\rho, \alpha)), a) \\
&= \mathsf{q}_{A[\sigma \circ \nu]}\, (\rho, s, \alpha, a)
\end{aligned}$$

The semantic counterparts of $\mathsf{U}[]$ and $\mathsf{El}[]$ are verified as follows.

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

For reasons of space, we only state what we need to verify for Π[]. It is enough to show that the mapping parts of the unquoted functions are equal and that the first components of the results of quote are equal because the other parts are equalities.

$$\mathsf{map}\left(\mathsf{u}_{(\Pi\,A\,B)[\sigma]}\,(\rho,n,\alpha)\right) \equiv \mathsf{map}\left(\mathsf{u}_{\Pi\,A[\sigma]\,B[\sigma^A]}\,(\rho,n,\alpha)\right)$$
$$\mathsf{proj}_1\left(\mathsf{q}_{(\Pi\,A\,B)[\sigma]}\,(\rho,t,\alpha,f)\right) \equiv \mathsf{proj}_1\left(\mathsf{q}_{\Pi\,A[\sigma]\,B[\sigma^A]}\,(\rho,t,\alpha,f)\right)$$

The methods for substitutions and terms (including the equality methods) are all trivial.

8 Normalisation

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

$$\mathsf{norm}_A\ \ (t : \mathsf{Tm}\,\Gamma\,A) : \mathsf{Nf}\,\Gamma\,A \qquad := \mathsf{proj}_1\left(\mathsf{q}_A\left(\mathsf{id}, {}_{[\mathsf{id}]^{-1}}{*}t, \mathsf{u}_\Gamma\,\mathsf{id}, [\![t]\!]\,(\mathsf{id}, \mathsf{u}_\Gamma\,\mathsf{id})\right)\right)$$
$$\mathsf{compl}_A\,(t : \mathsf{Tm}\,\Gamma\,A) : t \equiv \ulcorner \mathsf{norm}_A\,t \urcorner := \mathsf{proj}_2\left(\mathsf{q}_A\left(\mathsf{id}, {}_{[\mathsf{id}]^{-1}}{*}t, \mathsf{u}_\Gamma\,\mathsf{id}, [\![t]\!]\,(\mathsf{id}, \mathsf{u}_\Gamma\,\mathsf{id})\right)\right)$$

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

$$\frac{n : \mathsf{Ne}\,\Gamma\,A}{[\![\ulcorner n \urcorner]\!]\,(\mathsf{id}, \mathsf{u}_\Gamma\,\mathsf{id}) \equiv \mathsf{u}_A\,(\mathsf{id}, n, \mathsf{u}_\Gamma\,\mathsf{id})} \qquad \frac{n : \mathsf{Nf}\,\Gamma\,A}{\mathsf{norm}_A \ulcorner n \urcorner \equiv v}$$

As our normal forms are indexed by types, we need decidability of equality of types to show decidability of equality of normal forms. For this, we need to define a model of normal forms where types are mapped to normal types (which exclude substituted types). We leave this as future work.

9 Conclusions and further work

We proved normalisation for a basic type theory with dependent types by the technique of NBE. We evaluate terms into a proof relevant logical predicate model. The model is depending on the syntax, we need to use the dependent eliminator of the syntax. Our approach can be seen as merging the presheaf model and the logical relation used in NBE for simple types [6] into a single logical predicate. This seems to be necessary because of the combination of type indexing and dependent types: the well-typedness of normalisation depends on completeness. Another property to note is that we don't normalise types, we just index normal terms by not necessarily normal types.

We are currently working on completing the formalisation[6] [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 missing computation rules, reasoning with them involves lots of boilerplate. We expect that a cubical metatheory [12] with its systematic way of expressing equalities depending on equalities and its additional computation rules would significantly reduce the amount of boilerplate.

Another challenge is to extend our basic type theory with 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 K). We would also like to investigate whether the logical predicate interpretation can be generalised to work over arbitrary presheaf models and how the syntactic model fits here.

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

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments and suggestions.

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. *Normalization by Evaluation: Dependent Types and Impredicativity.* PhD thesis, Habilitation, Ludwig-Maximilians-Universität München, 2013.

4 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.

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. 1997.

9 Thorsten Altenkirch and Ambrus Kaposi. Agda formalisation 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.

11 Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed λ-calculus. In *Logic in Computer Science, 1991. LICS'91., Proceedings of Sixth Annual IEEE Symposium on*, pages 203–211. IEEE, 1991.

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

13 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.

14 Olivier Danvy. *Type-directed partial evaluation.* Springer, 1999.

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

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

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

18 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.