Extensional Equality in Intensional Type Theory

Thorsten Altenkirch
Department of Informatics
University of Munich
Oettingenstr. 67, 80538 München, Germany,
alti@informatik.uni-muenchen.de

Abstract

We present a new approach to introducing an extensional propositional equality in Intensional Type Theory. Our construction is based on the observation that there is a sound, intensional setoid model in Intensional Type theory with a proof-irrelevant universe of propositions and η-rules for Π- and Σ-types. The Type Theory corresponding to this model is decidable, has no irreducible constants and permits large eliminations, which are essential for universes.
Keywords. *Type Theory, categorical models.*

1. Introduction and Summary

In Intensional Type Theory (see e.g. [11]) we differentiate between a decidable definitional equality (which we denote by $=$) and a propositional equality type ($\mathrm{Id}_\sigma(-,-)$ for any given type σ) which requires proof. Typing only depends on definitional equality and hence is decidable.

In Intensional Type Theory the type corresponding to the principle of extensionality

$$\begin{aligned}\mathrm{Ext}_{x\in\sigma.\tau(x)} \quad\equiv\quad & \Pi_{f,g\in(\Pi x\in\sigma.\tau(x))} \\ & \quad(\Pi_{x\in\sigma}\mathrm{Id}_{\tau(x)}(f(x),g(x))) \\ & \quad\to \mathrm{Id}_{\Pi x\in\sigma.\tau(x)}(f,g)\end{aligned}$$

is not inhabited. To see this let $\sigma = \tau = \mathrm{Nat}$ and $f = \lambda x{\in}\mathrm{Nat}.x, g = \lambda x{\in}\mathrm{Nat}.x+0$, where addition is defined s.t. $x \neq x+0$. Using Nat-elimination we can show that $\Pi_{x\in\mathrm{Nat}}\mathrm{Id}_{\mathrm{Nat}}(f(x),g(x))$ is inhabited but $\mathrm{Id}_{\mathrm{Nat}\to\mathrm{Nat}}(f,g)$ is not, because f and g have different normal forms.

It has been an open problem how to extend Intensional Type Theory s.t. Ext is inhabited without destroying other fundamental properties. Ext is provable in Extensional Type Theory [10], where propositional and definitional equality are identified, but then equality and type checking become undecidable. We may introduce a constant of type Ext but then, using the elimination constant for Id, we can also define irreducible terms at other types (like Nat). We say that such a theory is not *adequate* because we can construct closed terms of type Nat which are not reducible to numerals.

The problem of extensionality in Intensional Type Theory has been extensively studied by Martin Hofmann [5]. His basic insight is that extensional equality can be modeled by an *intensional model construction* such as the setoid model, where every closed type is interpreted by a type and an equivalence relation. However, a naive version of the setoid model does not work because it does not satisfy all the required definitional equalities. In [6] Hofmann presents a solution using a modified interpretation of families. Unfortunately, this approach does not allow definitions of sets or propositions by recursion (large eliminations), in particular this prohibits the introduction of universes.

Our solution is also based on the setoid model but as the metatheory, where the construction takes place, we use an extension of Intensional Type Theory by a universe of propositions $\mathbf{Prop}$, such that all proofs of a proposition are definitionally equal. Moreover, we assume that the η-rules for Π-types and Σ-types (surjective pairing) hold in the metatheory. We show that this extension of Intensional Type Theory is decidable. Inside our metatheory we define an intensional model which corresponds to a Type Theory – the object theory – with the following properties:

1. Definitional equality (and hence typechecking) is decidable.
2. Ext is inhabited.
3. The theory is adequate, i.e. all closed natural numbers are definitionally equal to numerals.
4. Large eliminations (e.g. a simply typed universe) can be interpreted.

Acknowledgments. I would like to thank the following people for interesting discussions, comments on earlier drafts and proof-reading: Andreas Abel, Peter Dybjer, Martin Hofmann, Ralph Matthes, Bernhard Reus and Thomas Streicher.

2. The metatheory

We work in an Intensional Type Theory (e.g. see [7]) with $\mathbf{Set} \in \mathbf{Type}$ and $\mathbf{Set} \leq \mathbf{Type}$, i.e. every set is a type. We assume the existence of Π and Σ-types in $\mathbf{Set}$ and $\mathbf{Type}$.

We write the domain of a Π-type in subscript to signal implicit arguments which are omitted when applying a term of this type. If unambiguous we may overload a name (like the objects and homsets of a category). When introducing non-prefix operators we use "$-$" to mark the spaces for the explicit arguments. The curried application of t to $u_1, \ldots, u_n$ is written $t(u_1, \ldots, u_n)$. The elements of Σ-types are written as pairs $(a, b)_{x\in\sigma.\tau(x)} \in \Sigma x \in \sigma.\tau(x)$, for Σ-elimination we use the projections π_1, π_2. Complex Σ-types are introduced using named projections. The type annotations can be omitted if they can be inferred from the context.

The definitional equality includes the η-rules[1]:

$$\begin{array}{rcll} \lambda x \in \sigma.f(x) & = & f & \quad x \text{ not free in } f \\ (\pi_1(t), \pi_2(t)) & = & t & \end{array}$$

As already indicated in the introduction our construction hinges on the existence of a proof-irrelevant universe of propositions $\mathbf{Prop} \in \mathbf{Type}$ and $\mathbf{Prop} \leq \mathbf{Set}$. The intuition is that $\mathbf{Prop}$ contains only sets with at most one inhabitant. This is reflected by decreeing that any two proofs of a proposition are definitionally equal:

$$\frac{\Gamma \vdash P \in \mathbf{Prop} \qquad \Gamma \vdash p, q \in P}{\Gamma \vdash p = q \in P}\ \text{proof} - \text{irr}$$

We introduce $\top, \bot \in \mathbf{Prop}$ as basic propositions together with the constructor $* \in \top$ and the absurdity elimination constant $\bot^e_\sigma \in \bot \to \sigma$ for any type σ. Furthermore we close $\mathbf{Prop}$ under Π and Σ-Types:

$$\frac{\Gamma \vdash A \in \mathbf{Set} \qquad \Gamma, x \in A \vdash P \in \mathbf{Prop}}{\Gamma \vdash \Pi x \in A.P \in \mathbf{Prop}}\ \Pi - \text{Prop}$$

$$\frac{\Gamma \vdash P \in \mathbf{Prop} \qquad \Gamma, x \in P \vdash Q \in \mathbf{Prop}}{\Gamma \vdash \Sigma x \in P.Q \in \mathbf{Prop}}\ \Sigma - \text{Prop}$$

Note that the domain of a propositional Π-type may be a set, we will denote this specific instance of a Π-type by $\forall x \in A.P$. If $P, Q \in \mathbf{Prop}$ and Q does not depend on P we write $P \wedge Q$ for $\Sigma x \in P.Q$. It is not necessary to introduce a propositional equality in the metatheory.

[1]We omit the assumptions that all the terms have the appropriate types.

We assume the existence of inductive types like the type of natural numbers $\mathrm{Nat} \in \mathbf{Set}$ with constructors $0 \in \mathrm{Nat}, \mathrm{s} \in \mathrm{Nat} \to \mathrm{Nat}$ and a family of elimination constants: Assume $\sigma(x) \in \mathbf{Type}$ for $x \in \mathrm{Nat}$

$$\begin{array}{rcl} \mathrm{R}^{\mathrm{Nat}}_\sigma & \in & \sigma(0) \\ & & \to (\Pi_{n\in\mathrm{Nat}}\sigma(n) \to \sigma(\mathrm{s}(n))) \\ & & \to \Pi_{n\in\mathrm{Nat}}\sigma(n) \end{array}$$

subject to the usual equations for primitive recursion.

We say that a Type Theory is decidable if definitional equality is decidable. This entails the decidability of type checking. A theory is consistent if it is logically consistent (not all types are inhabited) and equationally consistent (not all well typed definitional equalities hold). It is adequate if all closed terms of type Nat are definitionally equal to a numeral (i.e. a term of the form $\mathrm{s}^i(0)$).

Proposition 1 (Properties of the metatheory) *The extension of Intensional Type Theory by a universe of proof irrelevant propositions and η-rules described above is decidable, consistent and adequate.*

Proof (Sketch): It is well known that the standard β-reduction excluding the η-rules and proof-irr is terminating and Church-Rosser (e.g. see [1]). On the normal forms we introduce a structural equivalence which incorporates the η-rules and proof-irr. We show that this equivalence is decidable and that two well-typed terms are definitionally equal iff their normal forms are structurally equivalent. Hence $=$ is decidable. It is standard that the decidability of equality entails the decidability of type checking. Logical consistency follows from strong normalisation. Equational consistency can be derived from Church-Rosser and the fact that the congruence does not affect Nat, hence $0 \neq \mathrm{s}(0)$. Adequacy holds since the numerals are the only closed normal forms of type Nat. $\Box$

3. The object theory

In this section we shall specify the object theory, which represents our solution to the problem of extensionality. We summarize here the essential properties of this theory, which is an extension of basic type theory with Π-types (the logical framework or λ^Π) including the η-rule for Π-types. As a basic type we assume a type of natural numbers $\mathrm{Nat} : \mathbf{Type}$ with the constants and equations as defined in the previous section.

The object theory features an equality type

$$\mathrm{Id}_\sigma(x, y) : \mathbf{Type} \qquad [\sigma : \mathbf{Type}, x, y : \sigma]$$

with the constants

$$\begin{array}{rcl} \mathrm{refl}_\sigma & : & \Pi x:\sigma.\mathrm{Id}_\sigma(x,x) \\ \mathrm{subst}_{x:\sigma.\tau(x)} & : & \Pi_{x,y:\sigma}(\mathrm{Id}_\sigma(x,y)) \to \tau(x) \to \tau(y) \\ \mathrm{refl}^*_\sigma & : & \Pi_{x:\sigma}\Pi y:\tau(x). \\ & & \mathrm{Id}_{\tau(x)}(\mathrm{subst}_{x:\sigma.\tau(x)}(\mathrm{refl}_\sigma(x),y),y) \\ \mathrm{ext}_{x:\sigma.\tau(x)} & : & \mathrm{Ext}_{x:\sigma.\tau(x)} \end{array}$$

refl* corresponds to the β rule for the usual definition of intensional identity types. Here we require only that this equality is provable for propositional equality. The only definitional equality we postulate is that any two equality proofs are definitionally equal:

$$p = q \qquad [p, q : \mathrm{Id}_\sigma(t,u)]$$

To show that our construction works for large eliminations we include a simply typed universe, given by a type $\mathrm{U} : \mathbf{Type}$ and a family $\mathrm{El}(a) : \mathbf{Type}$ for any $a : \mathrm{U}$ with the following constants:

$$\begin{array}{rcl} \mathrm{nat} & : & \mathrm{U} \\ \mathrm{arr} & : & \mathrm{U} \to \mathrm{U} \to \mathrm{U} \end{array}$$

s.t. the following equalities hold:

$$\begin{array}{rcl} \mathrm{El}(\mathrm{nat}) & = & \mathrm{Nat} \\ \mathrm{El}(\mathrm{arr}(a,b)) & = & \mathrm{El}(A) \to \mathrm{El}(B) \end{array}$$

Our goal is to verify that such a Type Theory exists and is also decidable, consistent and adequate. The common approach to introduce Type Theories is syntactical: decidability can be verified by a combination of a Church-Rosser-theorem and strong normalisation. Our impression is that the approach fails here.

As indicated in the introduction we shall follow a different path here: we define a model of Type Theory inside our type theoretic metatheory and verify that it is has all the required properties. This can be summarized in the following theorem:

Proposition 2 (Existence of object theory) *There is a model of type theory with constants and equalities interpreting the object theory defined above, which is decidable, consistent and adequate.*

4. The model construction

The notion of model we are using are *categories with families* as introduced by Dybjer and Hofmann [3, 7]. This is an intensional notion of model, i.e. definitional equalities in the object theory are interpreted by definitional equalities in the metatheory. Such intensional models always give rise to a decidable theory since definitional equality in the metatheory is decidable.

We give a detailed presentation of *categories with families* in appendix A. The basic idea is to define a category (Con) of semantic contexts and context morphisms. Semantic types Ty and terms Tm can be interpreted as a functor from Con to the category of families of sets s.t. Ty is the first (set) component of this functor and Tm the second (family) component. We then show how to interpret the empty context, context extension and Π-types. The interpretation of the syntax and the soundness theorem are given in [7], section 3.5.

It is cumbersome to check all the details of the model construction, which we will only sketch here. Hence we found it useful to employ the LEGO system [9]. One problem we faced is that LEGO (or any other implementation of Type Theory) does not implement the metatheory described in section 2. As a workaround we use LEGO with additional constants which η-expand elements of Π and Σ-types and map all elements of propositional types to a canonical constant. All those terms are identities in our intended metatheory. We allow ourselves to decorate any type with those expansion terms. See [2] for a partial implementation of the model in LEGO.

4.1. The category of setoids

The basic concept of the model construction are setoids, i.e. sets with an equivalence relation. We shall use setoids to interpret contexts hence we define $X \in \mathrm{Con}$ by the following structure (i.e. a Σ-type with named projections):

$$\begin{array}{rcl} X_{\mathrm{set}} & \in & \mathbf{Set} \\ - \sim_X - & \in & X \to X \to \mathbf{Prop} \\ X_{\mathrm{refl}} & \in & \forall_{x \in X_{\mathrm{set}}} x \sim_X x \\ X_{\mathrm{sym}} & \in & \forall_{x,y \in X_{\mathrm{set}}} (x \sim_X y) \to (y \sim_X x) \\ X_{\mathrm{trans}} & \in & \forall_{x,y,z \in X_{\mathrm{set}}} \\ & & \quad (x \sim_X y) \to (y \sim_X z) \to (x \sim_X z) \end{array}$$

In the sequel we shall often omit the index of $\sim$ if it is obvious which setoid is meant.

It is important to note that $\sim$ is propositional, hence we do not have to state any equalities regarding the inhabitants of $\sim$. Categorically any setoid is a trivial groupoid, i.e. a category where every morphism is an isomorphism.

Morphisms between setoids $f \in \mathrm{Con}(X,Y)$ correspond to substitutions in the syntax. They are given by functions between the sets which respect the equivalence relation:

$$\begin{array}{rcl} f_{\mathrm{fn}} & \in & X_{\mathrm{set}} \to Y_{\mathrm{set}} \\ f_{\mathrm{resp}} & \in & \forall_{x,y \in X_{\mathrm{set}}} (x \sim_X y) \to (f_{\mathrm{fn}}(x) \sim_Y f_{\mathrm{fn}}(y)) \end{array}$$

The definition of identity and composition and the verification of the categorical laws is straightforward (but requires the η-rules for Π and Σ-types). The fact that f_{resp} is propositional is not essential here, but simplifies the verification.

4.2. Types and terms

Since setoids are groupoids, it seems natural to define types over a fixed object $X \in \text{Con}$ as functors from X to Con. This is essentially the approach we take, but we only require the functor laws to hold up to the internal equivalence of setoids. Hence, a semantic type $A \in \text{Ty}(X)$ is given by the following structure:

$$\begin{array}{rcl}
A_{\text{fm}} & \in & X_{\text{set}} \to \text{Con} \\
A_{\text{subst}} & \in & \Pi_{x,y\in X_{\text{set}}}(x \sim_X y) \\
& & \to A_{\text{fm}}(x)_{\text{set}} \to A_{\text{fm}}(y)_{\text{set}} \\
A_{\text{subst}^*} & \in & \forall_{x,y\in X_{\text{set}},p\in x\sim_X y,a,b\in A_{\text{fm}}(x)_{\text{set}}}(a \sim b) \\
& & \quad \to (A_{\text{subst}}(p,a) \sim A_{\text{subst}}(p,b)) \\
A_{\text{refl}^*} & \in & \forall_{x\in X_{\text{set}},a\in A_{\text{fm}}(x)_{\text{set}}} \\
& & A_{\text{subst}}(X_{\text{refl}}(x),a) \sim a \\
A_{\text{trans}^*} & \in & \forall_{x,y,z\in X_{\text{set}},p\in(x\sim_X y),q\in(y\sim_X z),a\in A_{\text{fm}}(x)_{\text{set}}} \\
& & A_{\text{subst}}(q, A_{\text{subst}}(p,a)) \sim A_{\text{subst}}(X_{\text{trans}}(p,q),a)
\end{array}$$

A_{fm} is the object part of the functor, A_{subst} the morphism part, where A_{subst^*} corresponds to the condition that a morphism between setoids has to preserve the equivalence relation. A_{refl^*} and A_{trans^*} correspond to the functor laws up to the internal equivalence of setoids ($\sim$).

Semantic types form a presheaf, i.e. given $A \in \text{Ty}(Y)$ and a setoid morphism $f \in \text{Con}(X,Y)$ we can construct $A[f] \in \text{Ty}(X)$ s.t. identity and composition is preserved. This corresponds to substituting in a type syntactically.

$A[f]$ is given by

$$\begin{array}{rcl}
A[f]_{\text{fm}} & = & \lambda x \in X_{\text{set}}.A_{\text{fm}}(f_{\text{fn}}(x)) \\
A[f]_{\text{subst}} & = & \lambda x,y \in Y_{\text{set}}.\lambda p \in (x \sim_Y y). \\
& & \quad A_{\text{subst}}(f_{\text{resp}}(p)) \\
A[f]_{\text{subst}^*} & = & \lambda x,y \in Y_{\text{set}}, p \in x \sim_X y. \\
& & \quad A_{\text{subst}^*}(f_{\text{resp}}(p)) \\
A[f]_{\text{refl}^*} & = & \lambda x \in X_{\text{set}}.A_{\text{refl}^*}(f_{\text{fn}}(x)) \\
A[f]_{\text{trans}^*} & = & \lambda x,y,z \in X_{\text{set}}. \\
& & \lambda p \in (x \sim_X y), q \in (y \sim_X z). \\
& & \quad A_{\text{trans}^*}(f_{\text{resp}}(p), f_{\text{resp}}(q))
\end{array}$$

To see that $A[f]_{\text{refl}^*}$ and $A[f]_{\text{trans}^*}$ have the correct types we need that

$$f_{\text{resp}}(X_{\text{refl}}(x)) = X_{\text{refl}}(f_{\text{fn}}(x))$$

and

$$f_{\text{resp}}(X_{\text{trans}}(p,q)) = X_{\text{trans}}(f_{\text{resp}}(p), f_{\text{resp}}(q))$$

Both are instances of proof-irr since $- \sim - \in \mathbf{Prop}$.

The functor laws

$$\begin{array}{rcl}
A[1_Y] & = & A \\
A[f \circ g] & = & A[f][g]
\end{array}$$

can be verified automatically exploiting the fact that $=$ is decidable.

Given a type $A \in \text{Ty}(X)$ in a context X we define the set of semantic terms $t \in \text{Tm}(X,A)$, corresponding to the judgments $\Gamma \vdash t : \sigma$. Since types correspond to functors terms correspond to global elements of a functor, i.e. to families of elements in $A_{\text{fm}}(x)_{\text{set}}$ indexed by $x \in X_{\text{set}}$ which respect the equality. This can be spelt out as follows:

$$\begin{array}{rcl}
t_{\text{tm}} & \in & \Pi x \in X_{\text{set}}.A_{\text{fm}}(x)_{\text{set}} \\
t_{\text{resp}} & \in & \forall_{x,y\in X_{\text{set}},p\in(x\sim_X y)} \\
& & \quad A_{\text{subst}}(p, t_{\text{tm}}(x)) \sim_{A_{\text{fm}}(y)} t_{\text{tm}}(y)
\end{array}$$

Note that we have to use A_{subst} to define what we mean by *preserves equality*, since the images of equal elements in X end up in different slices of A_{fm}. As for types we also have to define the effect of substitutions on terms, i.e. given $t \in \text{Tm}(X,A), f \in \text{Con}(Y,X)$ we define

$$\begin{array}{rcl}
t[f] & \in & \text{Tm}(Y, A[f]) \\
t[f]_{\text{tm}} & = & \lambda x.t_{\text{tm}}(f_{\text{fn}}(x)) \\
t[f]_{\text{resp}} & = & \lambda x,y,p.t_{\text{resp}}(f_{\text{resp}}(p))
\end{array}$$

We also have to check that this construction preserves identity and composition:

$$\begin{array}{rcl}
t[1_Y] & = & t \\
t[f \circ g] & = & t[f][g]
\end{array}$$

Note that these equations are only well typed if the presheaf laws for $A[f]$ hold. Again they can be checked automatically, see [2].

4.3. Contexts

We have to define an interpretation of the empty context $\bullet \in \text{Con}$:

$$\begin{array}{rcl}
\bullet_{\text{set}} & = & \top \\
x \sim_\bullet y & = & \top
\end{array}$$

This is a terminal object in Con.

Given a context $X \in \text{Con}$ and a type $A \in \text{Ty}(X)$ we can construct a new context $X.A$ which syntactically corresponds to introducing a new variable of type $A \in \text{Ty}(X)$. The non-propositional components of $X.A$ are given by:

$$\begin{array}{rcl}
(X.A)_{\text{set}} & = & \Sigma x \in X_{\text{set}}.A_{\text{fm}}(x)_{\text{set}} \\
(x,a) \sim_{X.A} (y,b) & = & \Sigma p \in (x \sim_X y). \\
& & \quad A_{\text{subst}}(p,a) \sim_{A_{\text{fm}}(y)} b
\end{array}$$

Note that Σ-Prop in its dependent form is essential to type $\sim_{X.A}$.

We have to show that $\sim_{X.A}$ is an equivalence relation. Reflexivity follows from X_{refl} and A_{refl^*}. Transitivity can be shown similarly using X_{trans} and A_{trans^*}. To show symmetry assume $(p,q) \in (x,a) \sim_{X.A} (y,b)$, i.e. $p \in x \sim y$ and $q \in A_{\mathrm{subst}}(p,a) \sim b$. Clearly $p' = X_{\mathrm{sym}}(p) \in y \sim x$, we have to construct $q' \in A_{\mathrm{subst}}(p',b) \sim a$. This can be derived from $A_{\mathrm{fm}}(x)_{\mathrm{sym}}(q)$ using A_{refl^*} and A_{trans^*}.

We have to define projections fst and snd, a pairing operation $(-,-)$ and an empty substitution $()$. The projections are needed to interpret variables — snd is the last variable and fst corresponds to weakening. Their set-theoretic components are given by the obvious definitions:

$$\begin{aligned}
() &\in \Pi_{\Delta\in\mathrm{Con}}.\mathrm{Con}(\Delta,\bullet)\\
()_{\mathrm{fn}} &= \lambda x.*\\
(-,-) &\in \Pi_{\Gamma,\Delta\in\mathrm{Con},A\in\mathrm{Ty}(\Delta)}.\Pi f\in\mathrm{Con}(\Gamma,\Delta).\\
&\qquad \mathrm{Tm}_\Gamma(A[f]) \to \mathrm{Con}(\Gamma,\Delta.A)\\
(f,t)_{\mathrm{fn}} &= \lambda x.(f_{\mathrm{fn}}(x),t_{\mathrm{tm}}(x))\\
\mathrm{fst} &\in \Pi_{\Gamma,\Delta\in\mathrm{Con},A\in\mathrm{Ty}(\Delta)}.\\
&\qquad \mathrm{Con}(\Gamma,\Delta.A)\to\mathrm{Con}(\Gamma,\Delta)\\
\mathrm{fst}(f)_{\mathrm{fn}} &= \lambda x.\pi_1(f_{\mathrm{fn}}(x))\\
\mathrm{snd} &\in \Pi_{\Gamma,\Delta\in\mathrm{Con},A\in\mathrm{Ty}(\Delta)}.\\
&\qquad \Pi f\in\mathrm{Con}(\Gamma,\Delta.A).\mathrm{Tm}_\Gamma(A[\mathrm{fst}(f)])\\
\mathrm{snd}(f)_{\mathrm{tm}} &= \lambda x.\pi_2(f_{\mathrm{fn}}(x))
\end{aligned}$$

The verification of the corresponding resp components is straightforward.

Using pairing and projection we can define a lifting operation which will be useful for the definition of Π-types: given $f\in\mathrm{Con}(X,Y), A\in\mathrm{Ty}(Y)$ we define

$$\begin{aligned}
f^A &\in \mathrm{Con}(X.A[f],Y.A)\\
&= (f\circ\mathrm{fst}(1_{X.A[f]}),\mathrm{snd}(1_{X.A[f]}))
\end{aligned}$$

We note that

$$f^A_{\mathrm{fn}}(x,a) \quad = \quad (f_{\mathrm{fn}}(x),a)$$

4.4. Π-types

Higher order types, i.e. Π-types, provide the essential test that our construction works. Given $A\in\mathrm{Ty}(X)$ and a semantic type $B\in\mathrm{Ty}(A.X)$ we define $\Pi(A,B)\in\mathrm{Ty}(X)$. This corresponds to the Π-formation rule. Elements of Π-types are dependent functions which respect the equivalence relation, i.e. assuming $x\in X_{\mathrm{set}}$: $f\in\Pi(A,B)_{\mathrm{fm}}(x)_{\mathrm{set}}$ is given by the following structure:

$$\begin{aligned}
f_{\mathrm{fn}} &\in \Pi a\in A_{\mathrm{fm}}(x)_{\mathrm{set}}.B_{\mathrm{fm}}(x,a)_{\mathrm{set}}\\
f_{\mathrm{resp}} &\in \forall_{a,b\in A_{\mathrm{fm}}(x)_{\mathrm{set}},p\in(a\sim_{A_{\mathrm{fm}}(x)}b)}\\
&\qquad B_{\mathrm{subst}}((X_{\mathrm{refl}}(x),p),f_{\mathrm{fn}}(a))\sim f_{\mathrm{fn}}(b)
\end{aligned}$$

The associated equality is pointwise equality:

$$f\sim_{\Pi(A,B)_{\mathrm{fm}}(x)} g \quad = \quad \forall a\in A_{\mathrm{fm}}(x)_{\mathrm{set}}.f_{\mathrm{fn}}(a)\sim g_{\mathrm{fn}}(a)$$

To show that $\sim_{\Pi(A,B)_{\mathrm{fm}}(x)}$ is an equivalence relation we exploit the corresponding conditions for $\sim_{B_{\mathrm{fm}}(x,a)}$.

To define $\Pi(A,B)_{\mathrm{subst}}$ assume $(f_{\mathrm{fn}},f_{\mathrm{resp}})\in\Pi(A,B)_{\mathrm{fm}}(x)$ and $p\in x\sim y$. We have to construct

$$\begin{aligned}
\Pi(A,B)_{\mathrm{subst}}(p,(f_{\mathrm{fn}},f_{\mathrm{resp}})) &= (g_{\mathrm{fn}},g_{\mathrm{resp}})\\
&\in \Pi(A,B)_{\mathrm{fm}}(y)
\end{aligned}$$

We define

$$\begin{aligned}
g_{\mathrm{fn}} &\in \Pi a\in A_{\mathrm{fm}}(y).B_{\mathrm{fm}}(y,a)\\
g_{\mathrm{fn}} &= \lambda a.B_{\mathrm{subst}}(p,p')(f_{\mathrm{fn}}(A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),a)))
\end{aligned}$$

where $p'\in A_{\mathrm{subst}}(p,A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),a))\sim a$ can be derived using A_{trans^*} and A_{refl^*}. To construct g_{resp} assume $q\in a\sim_{A_{\mathrm{fm}}(y)} b$, we have to show that

$$B_{\mathrm{subst}}((X_{\mathrm{refl}}(y),q),g_{\mathrm{fn}}(a))\sim g_{\mathrm{fn}}(b) \qquad \text{(Q)}$$

is inhabited. Let $a' = A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),a)$, $b' = A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),b)$. Using A_{subst^*} and f_{resp} we can derive

$$B_{\mathrm{subst}}((X_{\mathrm{refl}}(y),q),f_{\mathrm{fn}}(a'))\sim f_{\mathrm{fn}}(b') \qquad \text{(H)}$$

from q. This equality lives in $B_{\mathrm{fm}}(x,A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),b))$. We can construct $r\in(x,A_{\mathrm{subst}}(X_{\mathrm{sym}}(p),b)\sim(y,b)$ using A_{trans^*} and A_{refl^*}. Applying B_{subst^*} to r to (H) we derive an equation in the same slice as (Q). (Q) can be derived from this using B_{trans^*} and B_{refl^*} exploiting proof-irr.

The details of this construction and the derivation of subst^*, refl^* and trans^* have been formally checked, see [2].

We have to check that our definition of Π-types is consistent with substitution, this corresponds to the *Beck-Chevalley* condition in fibered category theory, i.e. given $f\in\mathrm{Con}(Y,X)$ we have to verify:

$$\Pi(A,B)[f] \quad = \quad \Pi(A[f],B[f^A])$$

Again this is an instance of the decidable definitional equality.

To interpret λ-abstraction and application in the syntax we have to provide the constants app and lam. Their tm components are given by:

$$\begin{aligned}
\mathrm{lam} &\in \Pi_{\Gamma\in\mathrm{Con},A\in\mathrm{Ty}(\Gamma),B\in\mathrm{Ty}(\Gamma.A)}.\\
&\qquad \mathrm{Tm}_{\Gamma.A}(B)\to\mathrm{Tm}_\Gamma(\Pi(A,B))\\
\mathrm{lam}(t)_{\mathrm{tm}} &= \lambda x\in X_{\mathrm{set}},a\in A_{\mathrm{fm}}(x)_{\mathrm{set}}.t_{\mathrm{tm}}((x,a))\\
\mathrm{app} &\in \Pi_{\Gamma\in\mathrm{Con},A\in\mathrm{Ty}(\Gamma),B\in\mathrm{Ty}(\Gamma.A)}.\\
&\qquad \mathrm{Tm}_\Gamma(\Pi(A,B))\to\mathrm{Tm}_\Gamma.A(B)\\
\mathrm{app}(t)_{\mathrm{tm}} &= \lambda x\in(X.A)_{\mathrm{set}}.f_{\mathrm{tm}}(x_1,x_2)
\end{aligned}$$

It is straightforward to verify the resp components. We also have to check the definitional equalities corresponding to β and η hold:

$$\begin{aligned} \mathrm{lam}(\mathrm{app}(u)) &= u \\ \mathrm{app}(\mathrm{lam}(t)) &= t \end{aligned}$$

and that both operations behave correctly wrt. substitution

$$\begin{aligned} \mathrm{lam}(t)[f] &= \mathrm{lam}(t[f^A]) \\ \mathrm{app}(u)[f^A] &= \mathrm{app}(u[f]) \end{aligned}$$

Non-dependent function spaces are a special case of the Π-construction. Given $A, B \in \mathrm{Ty}(X)$ we define

$$\begin{aligned} A \Rightarrow B &= \Pi(A, B[\mathrm{fst}]) \\ &\in \mathrm{Ty}(X) \end{aligned}$$

We note that $(A \Rightarrow B)\mathrm{fm}(x)$ can be defined using only $A_{\mathrm{fm}}(x), B_{\mathrm{fm}}(x)$ and we denote this by

$$- \Rightarrow_{\mathrm{fm}} - \;\in\; \mathrm{Con} \times \mathrm{Con} \to \mathrm{Con}$$

This observation leads to a simpler definition of the simply typed universe.

4.5. Natural numbers

Given $X \in \mathrm{Con}$ we define $[\![\mathrm{Nat}]\!] \in \mathrm{Ty}(X)$ by a constant family:

$$[\![\mathrm{Nat}]\!]_{\mathrm{fm}}(x)_{\mathrm{set}} = \mathrm{Nat}$$

$\sim_{[\![\mathrm{Nat}]\!]_{\mathrm{fm}}(x)}$ is defined by structural recursion, i.e. by translating the following definition into applications of $\mathrm{R}^{\mathrm{Nat}}_{\sigma}$:

$$\begin{aligned} 0 \sim 0 &= \top \\ \mathrm{s}(m) \sim 0 &= \bot \\ 0 \sim \mathrm{s}(n) &= \bot \\ \mathrm{s}(m) \sim \mathrm{s}(n) &= m \sim n \end{aligned}$$

Again using $\mathrm{R}^{\mathrm{Nat}}_{\sigma}$ it is straightforward to show that $\sim_{[\![\mathrm{Nat}]\!]_{\mathrm{fm}}(x)}$ is an equivalence relation.

The definition of $[\![0]\!]$ and $[\![\mathrm{s}]\!]$ is obvious:

$$\begin{aligned} [\![0]\!] &\in [\![\mathrm{Nat}]\!] \\ [\![0]\!]_{\mathrm{fn}} &= \lambda x.0 \\ [\![s]\!] &\in [\![\mathrm{Nat}]\!] \Rightarrow [\![\mathrm{Nat}]\!] \\ [\![s]\!]_{\mathrm{fn}} &= \lambda x, n.\mathrm{s}(n) \end{aligned}$$

The corresponding resp-components are inhabited since $0 \sim 0$ and $\mathrm{s}(m) \sim \mathrm{s}(n) = m \sim n$.

The interpretation of $\mathrm{R}^{\mathrm{Nat}}_{\sigma}$ is more involved using the corresponding constant in the metatheory.

4.6. The simply typed universe

Given $X \in \mathrm{Con}$ we define $U \in \mathrm{Ty}(X)$ as a constant type similar to $[\![\mathrm{Nat}]\!]$ — $[\![U]\!]^0 = [\![U]\!]_{\mathrm{fm}}(x)_{\mathrm{set}}$ is given by the inductive type generated by

$$\frac{}{\mathrm{nat} \in [\![U]\!]^0} \qquad \frac{a, b \in [\![U]\!]^0}{\mathrm{arr}(a,b) \in [\![U]\!]^0}$$

We define $\sim_{[\![U]\!]} \in [\![U]\!]^0 \to [\![U]\!]^0 \to \mathbf{Prop}$ in the same fashion as $\sim_{\mathrm{Nat}}$:

$$\begin{aligned} \mathrm{nat} \sim_{[\![U]\!]} \mathrm{nat} &= \top \\ \mathrm{arr}(a,b) \sim_{[\![U]\!]} \mathrm{arr}(a',b') &= (a \sim_U a') \wedge (b \sim_U b') \\ a \sim_U a' &= \bot \qquad \text{otherwise} \end{aligned}$$

We define $[\![\mathrm{El}]\!] \in \mathrm{Ty}(X.[\![U]\!])$: $[\![\mathrm{El}]\!]_{\mathrm{fm}}$ is given by:

$$\begin{aligned} [\![\mathrm{El}]\!]_{\mathrm{fm}}(x, \mathrm{nat}) &= [\![\mathrm{Nat}]\!]_{\mathrm{fm}}(x) \\ [\![\mathrm{El}]\!]_{\mathrm{fm}}(x, \mathrm{arr}(a,b)) &= [\![\mathrm{El}]\!]_{\mathrm{fm}}(x,a) \Rightarrow_{\mathrm{fm}} [\![\mathrm{El}]\!]_{\mathrm{fm}}(x,b) \end{aligned}$$

$[\![\mathrm{El}]\!]_{\mathrm{subst}}$ and the other components of the structure can be derived from the corresponding components of $\Rightarrow$.

4.7. The equality type

Given $X \in \mathrm{Con}, A \in \mathrm{Ty}(X)$ we define $[\![\mathrm{Id}]\!](A) \in \mathrm{Ty}(X.A.A^+)$, where $A^+ = A[\mathrm{fst}]$. From now in we shall ignore weakening morphisms and use variables to denote projections. We define:

$$\begin{aligned} [\![\mathrm{Id}]\!](A)_{\mathrm{fm}}(x,a,b)_{\mathrm{set}} &= a \sim_{A_{\mathrm{fm}}(x)} b \\ p \sim_{[\![\mathrm{Id}]\!](A)_{\mathrm{fm}}(x,a,b)} q &= \top \end{aligned}$$

To define $[\![\mathrm{Id}]\!](A)_{\mathrm{subst}}$ assume that

$$(p,q,r) \in (x,a,b) \sim_{X.A.A^+} (y,a',b')$$

i.e. $p_x \in x \sim x'$,$p_a \in A_{\mathrm{subst}}(p,a) \sim a'$ and $p_b \in A_{\mathrm{subst}}(p,b) \sim b'$. Now given $q \in a \sim b$ we know $A_{\mathrm{subst}}(p,a) \sim A_{\mathrm{subst}}(p,b)$ using A^*_{subst}. Using $A_{\mathrm{fm}}(x)_{\mathrm{trans}}$ we can derive $a' \sim b'$. The verification of the other components of $[\![\mathrm{Id}]\!]$ is trivial because $\sim_{[\![\mathrm{Id}]\!]}$ is propositional.

To define $[\![\mathrm{refl}]\!]$ we construct

$$\begin{aligned} [\![\mathrm{refl}]\!]^0 &\in \mathrm{Tm}(x : X.a : A, [\![\mathrm{Id}]\!][(x,a,a)] \\ [\![\mathrm{refl}]\!]^0_{\mathrm{fn}} &= \lambda x, a.A_{\mathrm{fm}}(x)_{\mathrm{refl}} \\ [\![\mathrm{refl}]\!] &= \mathrm{lam}([\![\mathrm{refl}]\!]^0) \end{aligned}$$

In this case resp is trivial.

Given $B \in \mathrm{Ty}(X.A)$ we define

$$\begin{array}{rcl} [\![\mathrm{subst}]\!](A,B)^0 & \in & \mathrm{Tm}(x : X, a : A, b : A. \\ & & [\![\mathrm{Id}]\!][(x,a,b)].B[(x,a)], B[(x,b)]) \\ [\![\mathrm{subst}]\!](A,B)^0_{\mathrm{fn}} & = & \lambda(x,a,b,q,y). \\ & & \quad B_{\mathrm{subst}}((\mathrm{refl}(x),q),y) \\ [\![\mathrm{subst}]\!](A,B) & = & \mathrm{lam}^3([\![\mathrm{subst}]\!](A,B)^0) \end{array}$$

where lam^i is the i-fold application of lam.

We also define

$$\begin{array}{rcl} [\![\mathrm{refl}^*]\!](A,B)^0 & \in & \mathrm{Tm}(x : X.a : A.y : B[(x,a)], \\ & & [\![\mathrm{Id}]\!][(x,([\![\mathrm{subst}]\!](A,B)^0[(x,a,a, \\ & & \quad [\![\mathrm{refl}]\!]^0[(x,a)],y)],y))]) \\ [\![\mathrm{refl}^*]\!](A,B)^0_{\mathrm{fn}} & = & \lambda(x,a,y).B_{\mathrm{refl}^*}((x,a),y) \\ [\![\mathrm{refl}^*]\!](A,B) & = & \mathrm{lam}^2([\![\mathrm{refl}^*]\!](A,B)^0) \end{array}$$

Again resp is trivial.

Assuming $f,g \in \Pi(A,B)_{\mathrm{fm}}(x)_{\mathrm{set}}$ we define

$$\mathrm{HYP}(f,g) = \Pi(a : A[x], [\![\mathrm{Id}]\!][x,(\mathrm{app}(f,a),\mathrm{app}(g,a))]$$

We note that

$$\mathrm{HYP}(f,g)_{\mathrm{fm}}(x)_{\mathrm{set}} = [\![\mathrm{Id}]\!](\Pi(A,B))_{\mathrm{fm}}(x)_{\mathrm{set}}$$

Hence we can define

$$\begin{array}{rcl} [\![\mathrm{ext}]\!](A,B)^0 & \in & \mathrm{Tm}(x : X.f,g : \Pi(A,B)_{\mathrm{fm}}(x)_{\mathrm{set}}. \\ & & \quad q \in \mathrm{HYP}(f,g), \\ & & \quad\quad [\![\mathrm{Id}]\!](\Pi(A,B))[(x,f,g)]) \\ [\![\mathrm{ext}]\!](A,B)^0_{\mathrm{fn}} & = & \lambda(x,f,g,q).q \\ [\![\mathrm{ext}]\!](A,B) & = & \mathrm{lam}^3([\![\mathrm{ext}]\!](A,B)^0) \end{array}$$

All elements of $[\![\mathrm{Id}]\!](A)_{\mathrm{fm}}(x,a,b)_{\mathrm{set}}$ are definitionally equal in the metatheory since it is propositional, hence the required definitional equality holds.

4.8. Proof of the main theorem

Proof: The model construction above verifies our main theorem, proposition 2. We have interpreted all the constants introduced in section 3 and have checked that the equational conditions hold. Equality in the model is decidable because it is given by definitional equality in the metatheory. We observe that it is consistent since $\mathrm{Id}(0, s(0))$ is not inhabited and 0 and $s(0)$ are not definitionally equal. It is adequate because Nat only contains elements which are definitionally equal to numerals. Note that we use proposition 1 here.

5. Discussion and further work

To show that setoids from a category Con we do not require proof-irr, but already to have a notion of semantic types along the lines we have described here relies on this feature of the metatheory, e.g. see section 4.2. In the entire construction proof-irr is needed frequently.

We are not able to show that proof-irr is essential for the construction but experience with previous attempts (by the author and by Martin Hofmann) does suggest this. Having to deal with inconvertible proofs perpetrates the construction and eventually leads to failure. Having proof-irr adds some extensionality to the system, in particular since **Prop** is closed under Π-types whose domains are sets. In the pure system it is not even provable that a Π-type whose codomain is propositional, i.e. has at most one element, is propositional itself.

We have already mentioned Martin Hofmann's work on the subject [5],[6]. It is also interesting to compare our construction with the groupoid model used in [8]. Note, that the groupoid model requires an extensional type theory as metatheory. Another difference is that the equalities corresponding to $\mathrm{subst}^*, \mathrm{refl}^*$ and trans^* have to hold strictly (i.e. for metatheoretical equality) whereas we state them in terms of the equalities of the respective semantic types.

We have only presented a simply typed universe to show that our construction allows large eliminations and hence essentially generalizes Hofmann's construction [6]. We believe that it is possible to interpret a full dependent universe (corresponding to **Set** in our metatheory) using inductive-recursive definitions as introduced in [4], but we have not yet verified all the details.

It should be straightforward to interpret quotient types as described in [6]. Another interesting application is the introduction of coinductive types.

Finally, it would be interesting to implement the object theory directly, without translating it into the metatheory, possibly using a substitution calculus for dependent types.

References

[1] T. Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, University of Edinburgh, November 1993.

[2] T. Altenkirch. The implementation of a setoid model in LEGO. Available on the WWW at: `http://www.tcs.informatik.uni-muenchen.de/˜alti/drafts/setoid.html`, December 1998.

[3] P. Dybjer. Internal type theory. *Lecture Notes in Computer Science*, 1158, 1996.

[4] P. Dybjer. A general formulation of simultaneous inductive-recursive defi nitions in type theory. *Journal of Symbolic Logic*, 1997.

[5] M. Hofmann. *Extensional concepts in intensional type theory*. PhD thesis, University of Edinburgh, 1995.
[6] M. Hofmann. A simple model for quotient types. In *Proc. TLCA '95*, volume 902 of *LNCS*, pages 216–234, 1995.
[7] M. Hofmann. *Semantics of Logics of Computation*, chapter Syntax and Semantics of Dependent Types. Cambridge University Press, 1997.
[8] M. Hofmann and T. Streicher. The groupoid interpretation of type theory. In *Venice Festschrift*. 1996.
[9] Z. Luo and R. Pollack. The LEGO proof development system: A user's manual. LFCS report ECS-LFCS-92-211, University of Edinburgh, 1992.
[10] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
[11] T. Streicher. Investigations into intensional type theory, 1993. Habilitationsschrift.

A. Categories with families as models of Type Theory

A model of Type Theory is given by the following data:

A.1. A category of contexts and context morphisms

$$\begin{array}{rcl} \mathrm{Con} & \in & \mathbf{Type} \\ \mathrm{Con} & \in & \mathrm{Con} \to \mathrm{Con} \to \mathbf{Set} \\ 1 & \in & \Pi_{X\in\mathrm{Con}}.\mathrm{Con}(X,X) \\ -\circ- & \in & \Pi_{X_1,X_2,X_3\in\mathrm{Con}}.\mathrm{Con}(X_2,X_3) \\ & & \to \mathrm{Con}(X_1,X_2) \to \mathrm{Con}(X_1,X_3) \end{array}$$

which given $f \in \mathrm{Con}(Y,X), g \in \mathrm{Con}(Z_1,Y), h \in \mathrm{Con}(Z_2,Z_1)$ satisfies:

$$\begin{array}{rcl} 1_X \circ f & = & f \\ f \circ 1_Y & = & f \\ (f\circ g)\circ h & = & f\circ(g\circ h) \end{array}$$

A.2. A presheaf of types

$$\begin{array}{rcl} \mathrm{Ty} & \in & \mathrm{Con} \to \mathbf{Type} \\ -[-] & \in & \Pi_{X,Y\in\mathrm{Con}}.\mathrm{Ty}(Y) \\ & & \to \mathrm{Con}(X,Y) \to \mathrm{Ty}(X) \end{array}$$

which given $A \in \mathrm{Ty}(Y), f \in \mathrm{Con}(Z,Y), g \in \mathrm{Con}(X,Z)$ satisfies:

$$\begin{array}{rcl} A[1_Y] & = & A \\ A[f\circ g] & = & A[f][g] \end{array}$$

A.3. Families of terms

$$\begin{array}{rcl} \mathrm{Tm} & \in & \Pi_{X\in\mathrm{Con}}.\mathrm{Ty}(X) \to \mathbf{Set} \\ -[-] & \in & \Pi_{X,Y\in\mathrm{Con},A\in\mathrm{Ty}(Y)}.\mathrm{Tm}_Y(A) \\ & & \to \Pi f\in\mathrm{Con}(X,Y).\mathrm{Tm}_X(A[f]) \end{array}$$

which given $A \in \mathrm{Ty}(Y), t \in \mathrm{Tm}_Y(A), f \in \mathrm{Con}(Z,Y), g \in \mathrm{Con}(X,Z)$ satisfies

$$\begin{array}{rcl} t[1_Y] & = & t \\ t[f\circ g] & = & t[f][g] \end{array}$$

A.4. Wellfounded context comprehension

$$\begin{array}{rcl} \bullet & \in & \mathrm{Con} \\ -.- & \in & \Pi_{X\in\mathrm{Con}}.\mathrm{Ty}(X) \to \mathrm{Con} \\ () & \in & \Pi_{Y\in\mathrm{Con}}.\mathrm{Con}(Y,\bullet) \\ (-,-) & \in & \Pi_{X,Y\in\mathrm{Con},A\in\mathrm{Ty}(Y)}.\Pi f\in\mathrm{Con}(X,Y). \\ & & \mathrm{Tm}_X(A[f]) \to \mathrm{Con}(X,Y.A) \\ \mathrm{fst} & \in & \Pi_{X,Y\in\mathrm{Con},A\in\mathrm{Ty}(Y)}. \\ & & \mathrm{Con}(X,Y.A) \to \mathrm{Con}(X,Y) \\ \mathrm{snd} & \in & \Pi_{X,Y\in\mathrm{Con},A\in\mathrm{Ty}(Y)}. \\ & & \Pi f\in\mathrm{Con}(X,Y.A).\mathrm{Tm}_X(A[\mathrm{fst}(f)]) \end{array}$$

which given $e \in \mathrm{Con}(X,\bullet), f \in \mathrm{Con}(X,Y), A \in \mathrm{Ty}(Y) t \in \mathrm{Tm}_X(A[f]), h \in \mathrm{Con}(X,Y.A), g \in \mathrm{Con}(Z,X)$ satisfies:

$$\begin{array}{rcl} e & = & ()_X \\ \mathrm{fst}(f,t) & = & f \\ \mathrm{snd}(f,t) & = & t \\ (\mathrm{fst}(h),\mathrm{snd}(h)) & = & h \\ (f,t)\circ g & = & (f\circ g, t[g]) \\ \mathrm{fst}(f)\circ g & = & \mathrm{fst}(f\circ g) \\ \mathrm{snd}(h)[g] & = & \mathrm{snd}(h\circ g) \end{array}$$

Given $f \in \mathrm{Con}(X,Y), A \in \mathrm{Ty}(Y)$ define

$$\begin{array}{rcl} f^A & \in & \mathrm{Con}(X.A[f], Y.A) \\ & = & (f\circ\mathrm{fst}(1_{X.A[f]}), \mathrm{snd}(1_{X.A[f]})) \end{array}$$

A.5. Closure under Π

$$\begin{array}{rcl} \Pi & \in & \Pi_{X\in\mathrm{Con}}.\Pi A\in\mathrm{Ty}(X).\mathrm{Ty}(X.A) \to \mathrm{Ty}(X) \\ \mathrm{lam} & \in & \Pi_{X\in\mathrm{Con},A\in\mathrm{Ty}(X),B\in\mathrm{Ty}(X.A)}. \\ & & \mathrm{Tm}_{X.A}(B) \to \mathrm{Tm}_X(\Pi(A,B)) \\ \mathrm{app} & \in & \Pi_{X\in\mathrm{Con},A\in\mathrm{Ty}(X),B\in\mathrm{Ty}(X.A)}. \\ & & \mathrm{Tm}_X(\Pi(A,B)) \to \mathrm{Tm}_{X.A}(B) \end{array}$$

which given $A \in \mathrm{Ty}(X), B \in \mathrm{Ty}(X.A), f \in \mathrm{Con}(Y,X), t \in \mathrm{Tm}_{X.A}(B), u \in \mathrm{Tm}_X(\Pi(A,B))$ satisfies

$$\begin{array}{rcl} \Pi(A,B)[f] & = & \Pi(A[f], B[f^A]) \\ \mathrm{lam}(t)[f] & = & \mathrm{lam}(t[f^A]) \\ \mathrm{app}(u)[f^A] & = & \mathrm{app}(u[f]) \\ \mathrm{lam}(\mathrm{app}(u)) & = & u \\ \mathrm{app}(\mathrm{lam}(t)) & = & t \end{array}$$

B. Interpretation of the syntax

We define a partial interpretation ($[\![-]\!]$ of annotated syntax in a model as defined in the previous section, i.e. terms,types and substitutions are annotated with their contexts and types. We assume as given an interpretation of type constants $C \quad [\Gamma]$ as $[\![C]\!] \in \mathrm{Ty}([\![\Gamma]\!])$ and term constants $c : \sigma \quad [\Gamma]$ as $[\![c]\!] \in \mathrm{Tm}_{[\![\Gamma]\!]}([\![\sigma]\!])$.

Contexts $[\![\Gamma]\!] \in \mathrm{Con}$

$$\begin{array}{rcl} [\![]\!] & = & \bullet \\ [\![\Gamma.x:\sigma]\!] & = & [\![\Gamma]\!].[\![\sigma]\!] \end{array}$$

Variables $[\![x^{\Gamma,\sigma}]\!] \in \mathrm{Tm}([\![\Gamma]\!], [\![\sigma]\!])$

$$\begin{array}{rcl} [\![x^{\Gamma.x:\sigma,\sigma}]\!] & = & \mathrm{snd}(1_{[\![\Gamma.x:\sigma]\!]}) \\ [\![x^{\Gamma.y:\sigma,\tau}]\!] & = & [\![x^{\Gamma,\tau}]\!][\mathrm{fst}(1_{[\![\Gamma.x:\sigma]\!]})] \end{array}$$

Substitutions $[\![\vec{t}^{\Gamma,\Delta}]\!] \in \mathrm{Con}([\![\Gamma]\!], [\![\Delta]\!])$

$$\begin{array}{rcl} [\![]\!] & = & () \\ [\![\vec{t}, x = t]\!] & = & ([\![\vec{t}]\!], [\![t]\!]) \end{array}$$

Types $[\![\sigma^{\Gamma}]\!] \in \mathrm{Ty}([\![\Gamma]\!])$

$$[\![\Pi x:\sigma.\tau]\!] = \Pi([\![\sigma]\!], [\![\tau]\!])$$

Terms $[\![t^{\Gamma,\sigma}]\!] \in \mathrm{Tm}_{[\![\Gamma]\!]}([\![\sigma]\!])$

$$\begin{array}{rcl} [\![\lambda x:\sigma.t^{\tau}]\!] & = & \mathrm{lam}([\![t]\!]) \\ [\![t^{\Gamma,\Pi x:\sigma.\tau}(u^{\Gamma',\sigma'})]\!] & = & \mathrm{app}([\![t]\!])[(u, 1_{[\![\Gamma]\!]})] \\ & & \text{if } [\![\Gamma]\!] = [\![\Gamma']\!] \text{ and } [\![\sigma]\!] = [\![\sigma']\!] \end{array}$$

Given a definition of the following judgments:

Contexts	$\vdash \Gamma$
Substitutions	$\Gamma \vdash \vec{t} : \Delta$
Equality of substitutions	$\Gamma \vdash \vec{t} = \vec{u} : \Delta$
Types	$\Gamma \vdash \sigma$
Equality of types	$\Gamma \vdash \sigma = \tau$
Terms	$\Gamma \vdash t : \sigma$
Equality of terms	$\Gamma \vdash t = u : \sigma$

we can state the soundness theorem (see [7])

Theorem 1 (Soundness) *The partial interpretation given above is*

1. *total for derivable judgments,*
2. *derivable equalities are reflected by equal elements in the model.*