Elimination of extensionality in Martin-Löf type theory

Martin Hofmann

Department of Computer Science, University of Edinburgh
JCMB, KB, Mayfield Rd., Edinburgh EH9 3JZ, Scotland

Abstract. We construct a syntactic model of intensional Martin-Löf type theory in which two pointwise propositionally equal functions are propositionally equal themselves. In the model types are interpreted as types equipped with equivalence relations; the identity type at each type is interpreted as the associated relation. The interpretation function from the syntax to the model gives rise to a procedure which replaces all instances of the identity type by suitable relations defined by induction on the type structure and thereby eliminates instances of an axiom which states that pointwise propositionally equal functions are propositionally equal themselves. We also sketch how "quotient types" can be interpreted.

1 Martin-Löf's identity type

Martin-Löf introduced the *identity type* in order to internalise the notion of definitional equality [11]. For any two terms M, N of some type A a type $\mathrm{Eq}_A(M,N)$ is introduced, which should ideally be inhabited iff M and N are definitionally equal. This is achieved by the following rules.

$$\frac{\Gamma \vdash A}{\Gamma,\ x,y{:}A \vdash \mathrm{Eq}_A(x,y)}\ \textsc{Eq-Form}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathrm{refl}_A(M) : \mathrm{Eq}_A(M,M)}\ \textsc{Eq-Intro}$$

$$\frac{\begin{array}{l}\Gamma,\ x,y{:}A,\ p{:}\mathrm{Eq}_A(x,y) \vdash C(x,y,p)\\ \Gamma,\ x{:}A \vdash M : C(x,x,\mathrm{refl}_A(x))\end{array}}{\Gamma,\ x,y{:}A,\ p{:}\mathrm{Eq}_A(x,y) \vdash \mathrm{J}_C(M) : C(x,y,p)}\ \textsc{Eq-Elim-J}$$

$$\frac{\begin{array}{l}\Gamma,\ x{:}A,\ p{:}\mathrm{Eq}_A(x,x) \vdash C(x,p)\\ \Gamma,\ x{:}A \vdash M : C(x,\mathrm{refl}_A(x))\end{array}}{\Gamma,\ x{:}A,\ p{:}\mathrm{Eq}_A(x,x) \vdash \mathrm{K}_C(M) : C(x,p)}\ \textsc{Eq-Elim-K}$$

and equality rules

$$\begin{array}{ll}\mathrm{J}_C(M)(x,x,\mathrm{refl}_A(x)) = M(x) & \textsc{Eq-Comp-J}\\ \mathrm{K}_C(M)(x,\mathrm{refl}_A(x)) = M(x) & \textsc{Eq-Comp-K}\end{array}$$

Notation. Here and in the sequel we write $\Gamma \vdash A$ to express that A is a type in context Γ. Round brackets with commas are used to denote free variables and substitution in an informal way. So $\mathrm{Eq}_A(x, y)$ could also be written Eq_A if we do not want to emphasize the two particular free variables, and $\mathrm{Eq}_A(M, N)$ is a shorthand for $\mathrm{Eq}_A[x := M][y := N]$. We also use parentheses with informal pattern-matching for the definition of types and terms, so for example $C(x : A, y : B(x)) := \ldots$ defines a type C in the context $x : A, y : B(x)$. If the types A and B are clear from the context we also write $C(x, y) := \ldots$ or even $C(y) := \ldots$ since x can be inferred from the type of y. If several variables of one and the same type are declared in a context we write the type only once. So $x, y : A$ declares two variables of type A. We occasionally omit subscripts in type and term formers like Eq_A if they are clear from the context.

The eliminator J is the one originally used by Martin-Löf [11]. It expresses in an intensional way that the only elements of an identity type are those of the form $\mathrm{refl}_A(-)$: A predicate (C) over an identity type which holds for the canonical elements $\mathrm{refl}_S(-)$ holds everywhere. The homogeneous eliminator K has later been added by Streicher [13] with the aim to make any two elements of an identity type propositionally equal. The proof that with K any two terms of an identity type are propositionally equal is straightforward and may e.g. be found in [13]. One can show that this property fails in the pure theory without K. Various applications for K and metatheoretical properties of K can be found in loc. cit.

We say that two terms M, N of type A in context Γ are ***propositionally equal*** if there exists a term P such that

$$\Gamma \vdash P : \mathrm{Eq}_A(M, N)$$

is derivable. On the other hand if

$$\Gamma \vdash M = N : A$$

is derivable we call M and N ***definitionally equal***. This latter equality also makes sense for types.

Except in the empty context the propositional equality is strictly coarser than definitional equality. This means that if

$$\Gamma \vdash P : \mathrm{Eq}_A(M, N)$$

it is not necessarily the case that

$$\Gamma \vdash M = N : A$$

The reason is that P might have been obtained by induction on some free variable in Γ. If one forces the two notions of equality to agree by adding the ***equality reflection rule***

$$\frac{\Gamma \vdash P : \mathrm{Eq}_A(M, N)}{\Gamma \vdash M = N : A}$$

as is done in extensional type theory [9] then type checking becomes undecidable. Intuitively this is because the proof term P is lost upon application of this rule. For this and other reasons the equality reflection rule was later rejected by Martin-Löf. The identity type without the reflection *rule* is called *intensional.* It is the subject of this paper.

In the empty context equality reflection is admissible, though. Indeed if

$$\vdash P : \mathrm{Eq}_A(M, N)$$

then by strong normalisation we must have that P is of the form $\mathrm{refl}_A(\_)$ and thus $\vdash M = N$. This property is called *equality reflection principle.* It is commonly considered as the characteristic property of propositional identity [7] and is thus required to be respected by possible extensions to the theory. Below we will argue against the equality reflection principle and propose to "specify" propositional equality by a Leibniz principle.

1.1 The strength and the weakness of the intensional identity type

The above formulation of the identity type captures most of the rules governing definitional equality. Apart from reflexivity, symmetry, and transitivity, in particular the substitution rule for dependent types

$$\frac{\begin{array}{l}\Gamma, x : A \vdash B(x)\\ \Gamma \vdash M = N : A\\ \Gamma \vdash U : B(M)\end{array}}{\Gamma \vdash U : B(N)}\ \textsc{Dep-Subst}$$

is reflected in that from J we can define an operator Subst_B, such that if $M, N : A$ and $P : \mathrm{Eq}_A(M, N)$ then if $U : B(M)$ then $\mathrm{Subst}_B(P, U)$ is an element of type $B(N)$. We first define the type C with which J gets instantiated by

$$C(x, y{:}A,\ p{:}\mathrm{Eq}_A(x, y)) := B(x) \to B(y)$$

and the premise to J by

$$H(x : A) := \lambda u : B(x).u) : C(x, x, \mathrm{refl}_A(x))$$

Now we can define Subst_B as

$$\mathrm{Subst}_B(P, U) := \mathrm{J}_C(H)(M, N, P)\,U$$

In other words, in order to get a function from $B(x)$ to $B(y)$ in the presence of a proof $P{:}\mathrm{Eq}_A(x, y)$ it is enough to give an inhabitant of that type with x and y identified, and here the identity function does the job.

From the equality rule Eq-Comp-J we can deduce that

$$\mathrm{Subst}_B(\mathrm{refl}(M), U) = U$$

Moreover, using J and K we can show that these Subst functions are ***coherent***, i.e. they do not depend on the proof P and any diagram formed out of these functions commutes up to propositional equality

Also all the congruence rules for definitional equality hold for the identity type, with the exception of the ξ-rule

$$\frac{\Gamma,\ x{:}A \vdash M = N : B(x)}{\Gamma \vdash \lambda x{:}A.M = \lambda x{:}A.N : \Pi x{:}A.B}$$

which is not provable for propositional equality. This means that from a proof of pointwise equality of two (dependent) functions

$$\Gamma,\ x{:}A \vdash P(x) : \mathrm{Eq}_{B(x)}(F(x), G(x))$$

we cannot in general conclude their propositional equality, i.e. we cannot find an inhabitant of the type

$$\Gamma \vdash \mathrm{Eq}_{\Pi x:A.B(x)}(\lambda x{:}A.F(x), \lambda x{:}A.G(x))$$

The reason for this lies in the equality reflection *principle*, i.e. equality reflection in the empty context. If the ξ-rule held propositionally then e.g. from a proof

$$x : \mathrm{N} \vdash P(x) : \mathrm{Eq}_{\mathrm{N}}(F(x), G(x))$$

that F and G are pointwise equal functions on the natural numbers we could deduce an inhabitant of

$$\vdash \mathrm{Eq}_{\mathrm{N}\to\mathrm{N}}(\lambda x{:}\mathrm{N}.F(x), \lambda x{:}\mathrm{N}.G(x))$$

But then, since equality reflection holds in the empty context, we could conclude that F and G are definitionally equal. This is a contradiction because definitional equality is decidable and pointwise equality of functions is undecidable.

This argument not only shows that the propositional ξ-rule is not derivable, but also that any extension to the theory which includes the propositional ξ-rule must necessarily violate the equality reflection principle, i.e. there may be terms which are propositionally equal in the empty context without being definitionally equal.

We thus propose to give up the reflection principle as the defining property of propositional equality and instead to understand propositional equality as ***substitutability in every context*** in the sense that whenever M is propositionally equal to N and $C(M)$ is inhabited then so is $C(N)$ in a canonical way.

Now two pointwise equal functions are indeed substitutable for one another in every context except in one arising from the identity type itself. Clearly if $F(x)$ and $G(x)$ are pointwise propositionally equal and $C(f) = \mathrm{Eq}(f, \lambda x{:}A.F(x))$ then $C(\lambda x{:}A.F(x))$ is inhabited by reflexivity, but $C(\lambda x{:}A.G(x))$ is not, for this would entail the propositional ξ-rule.

So we propose to add a new family of constants to the theory

$$\frac{\Gamma, x : A \vdash P : \mathrm{Eq}_{B(x)}(F(x), G(x))}{\Gamma \vdash \mathrm{Ext}_{A,B}(P) : \mathrm{Eq}_{\Pi x:A.B(x)}(\lambda x : \sigma.F(x), \lambda x : \sigma.G(x))}$$

which both achieves (via Subst) substitutability of pointwise equal functions in every context, and repairs the just mentioned problem with the particular context $\mathrm{Eq}(-, \lambda x.G(x))$. This is essentially the solution proposed by Turner in [14]. Its serious drawback, immediately pointed out by Martin-Löf in a subsequent discussion also in [14] is that this introduces noncanonical elements in each type, since we have not specified, how the eliminators J and K should behave when applied to a proof having Ext as outermost constructor. For example, consider the (constant) family $f : A \to B \vdash \mathrm{N}$. Now if $x : A \vdash P(x) : \mathrm{Eq}(F\,x, G\,x)$ for two functions $F, G : \mathrm{N} \to \mathrm{N}$ then $\mathrm{Subst}\,(\mathrm{Ext}\,(P), 0)$ is an element of N in the empty context which does not reduce to canonical form. One might try to find suitable reduction rules for Ext under which this term would for example reduce to 0. Yet no satisfactory set of such rules has been found to date. Notice, however, that we can show that the term in question is *propositionally equal* to 0 because using J we can "replace" $\mathrm{Ext}\,(P)$ by an instance of refl.

In loc. cit. Turner proposes to add the (definitional) equation

$$\mathrm{Ext}\,(H) = \mathrm{refl}_{\Pi x:A.B(x)}(\lambda x : A.F(x))$$

where

$$H(x : A) := \mathrm{refl}_{B(x)}(F(x))$$

i.e. if we use Ext only to establish equality of definitionally equal terms then we may use refl straightaway. The equation is incomplete e.g. because even in its presence the above term is not equal to 0 or any other canonical natural number. As an example for the use of K we show how this equation holds *propositionally*. We instantiate K with

$$C(f : \Pi x{:}A.B(x),\ p{:}\mathrm{Eq}(f, f)) := \mathrm{Eq}(p, \mathrm{refl}(f))$$

and thus reduce the task of proving the equation for $P := \mathrm{Ext}\,(\ldots)$ to the task of proving it for $P := \mathrm{refl}(\ldots)$ in which case it is an instance of reflexivity. This property does not seem to be provable using J alone because the family we are eliminating over only makes sense for Eq restricted to the diagonal.

Let us now come back to the problem of the noncanonical elements introduced by Ext . The solution we are going to describe in this paper consists of a post-hoc translation of proofs containing Ext into ones in the pure theory in which every occurrence of the identity type will be replaced by an equality relation defined by induction on the type structure. On basic inductive types like N this relation will be the identity type itself, but for example on the type $\mathrm{N} \to \mathrm{N}$ it will be pointwise equality and so on.

This translation is performed by constructing a syntactic model for type theory including Ext in which types are types with relations and dependent types are dependent types together with dependent relations and substitution functions. The interpretation of the identity type in the model is basically the relation associated to each type, which is why the propositional ξ-rule holds in the model. Of course in the translation every functional variable in a context comes equipped with an additional assumption that it respects these relations, so contexts must be translated, too.

Given this translation we may view the type theory with Ext as a meta- or macro-language for a theory in which equality is defined along the type structure and in which every substitution must be validated by a tedious proof of substitutivity. Through the interpretation in the model (which can be implemented on a machine) these substitutivity proofs are generated automatically.

For the nondependent case and predicate logic this method is well-known, in [2] it is attributed to Gandy, cf. also [6]. The idea of interpreting type theory with extensionality assumptions in pure intensional type theory was put forward by Martin-Löf during the discussion in [14], but to the best of our knowledge this idea has never been pursued.

The paper is organised as follows. In the next section we look at the nondependent case first and then lay down the interpretation of contexts, dependent types, and terms. In sections 3 and 4 we describe the interpretation of the dependent product and of the intensional identity types for which the propositional ξ-rule holds. This intensional identity type comes in two versions. In the first one the equality rules EQ-COMP-J and EQ-COMP-K only hold propositionally. This means that although in the model the two terms on either side of these equations are not equal, the corresponding identity type receives a nonempty interpretation. In the second more complicated version the two equations do hold. With the simpler version we can thus interpret a theory where the rules EQ-COMP-J and EQ-COMP-K are replaced by constants witnessing their propositional companions.

Section 5 sketches the interpretation of other type formers in particular universes and quotient types. In some of these problems similar to those with the identity type occur. Certain equations only hold propositionally. So when doing proofs in the "macro-language" we have to accept that we sometimes have to use explicit instances of Subst where in the pure theory an automatic conversion by rule DEP-SUBST does the job.

Section 6 contains several conclusive remarks and mentions an existing Lego-implementation of the model.

2 The setoid model

We shall now describe the syntactic model in detail, and show that its structure suffices to interpret all of intensional type theory including Ext. The underlying syntactic system is intensional Martin-Löf type theory without universes as described in [11]. Our presentation is clearly influenced by "categorical type theory" as described e.g. in [4, 12]; we try, however, to avoid categorical terminology as far as possible and we assume (almost) no knowledge of category theory. For the *cognoscenti*: we construct a "category with attributes" in the sense of Cartmell [1], see also [5].

For expository reasons we start with the interpretation of nondependent types which later appear as special cases.

2.1 Setoids

Definition 1. A *setoid* S is a quadruple $(S_{\text{set}}, S_{\text{rel}}, S_{\text{sym}}, S_{\text{trans}})$ where S_{set} is a type in the empty context

$$\vdash S_{\text{set}}$$

together with a "binary relation" S_{rel}

$$s, s'{:}S_{\text{set}} \vdash S_{\text{rel}}(s, s')$$

which is proven symmetric and transitive by S_{sym} and S_{trans}

$$s, s'{:}S_{\text{set}},\ p{:}S_{\text{rel}}(s, s') \vdash S_{\text{sym}}(p) : S_{\text{rel}}(s', s)$$
$$s, s', s''{:}S,\ p{:}S_{\text{rel}}(s, s'),\ p'{:}S_{\text{rel}}(s', s'') \vdash S_{\text{trans}}(p, p') : S_{\text{rel}}(s, s'')$$

Two setoids are equal if all their four components are definitionally equal.

Examples of setoids. Every type T in the empty context gives rise to a setoid $\nabla(T)$ by $\nabla(T)_{\text{set}} := T$, $\nabla(T)_{\text{rel}}(t, t'{:}T) := \text{Eq}_T(t, t')$. If S and T are setoids then their product $S \times T$ is defined by $(S \times T)_{\text{set}} := S_{\text{set}} \times T_{\text{set}}$ and $(S \times T)_{\text{rel}}((s, t),\ (s', t')) := S_{\text{rel}}(s, s') \times T_{\text{rel}}(t, t')$. The definition of the remaining components is obvious.

Definition 2. A *morphism* from setoid S to setoid T is a pair $f = (f_{\text{fun}}, f_{\text{resp}})$ where

$$s{:}S_{\text{set}},\ \bar{s}{:}S_{\text{rel}}(s, s) \vdash f_{\text{fun}}(s, \bar{s}) : T_{\text{set}}$$

and

$$\begin{gathered} s{:}S_{\text{set}},\ \bar{s}{:}S_{\text{rel}}(s, s),\ s'{:}S_{\text{set}},\ \bar{s}'{:}S_{\text{rel}}(s', s'),\ p{:}S_{\text{rel}}(s, s') \vdash \\ f_{\text{resp}}(p) : T_{\text{rel}}(f_{\text{fun}}(s, \bar{s}), f_{\text{fun}}(s', \bar{s}')) \end{gathered}$$

Two morphisms are equal if their two components are definitionally equal.

Notice that the first component of a setoid morphism has two arguments. An element s of S_{set} and a proof $\bar{s} : S_{\text{rel}}(s, s)$. As a convention we always use overlined variable names for such reflexivity assumptions. The morphisms between S and T can be internalised by the following definition of function space.

$$(S \Rightarrow T)_{\text{set}} := \Pi s{:}S_{\text{set}}.S_{\text{rel}}(s, s) \rightarrow T_{\text{set}}$$

and

$$\begin{aligned} &(S \Rightarrow T)_{\text{rel}}(f, f') := \\ &\quad \Pi s{:}S_{\text{set}}.\Pi \bar{s}{:}S_{\text{rel}}(s, s).\Pi s'{:}S_{\text{set}}.\Pi \bar{s}'{:}S_{\text{rel}}(s', s'). \\ &\quad\quad S_{\text{rel}}(s, s') \rightarrow T_{\text{rel}}(f\ s\ \bar{s}, f'\ s'\ \bar{s}') \end{aligned}$$

Again we leave out the remaining components. By straightforward calculation we now obtain

Proposition 3. *If the underlying type theory has surjective pairing and an η-rule for the Π-type then the setoids form a cartesian closed category with terminal object.*

Thus setoids carry enough structure to interpret the simply typed λ-calculus. If surjective pairing and η-conversion are not available then we still get a model for λ-calculus without these rules. Our aim in the rest of this paper is to generalise this to a full model of Martin-Löf type theory in which the identity type at each setoid is interpreted as the corresponding relation.

2.2 Notation for contexts and telescopes

The contexts of setoids we are going to define are built upon ordinary syntactic contexts. We thus introduce some notation to describe various manoeuvres with contexts. We must make a compromise here between formal correctness and readability. A fully formal treatment would be possible only in terms of de Bruijn indices which we have not introduced.

The empty context is denoted $\underline{1}$. If $\Gamma \vdash A$ we write (Γ, A) for the context extension of Γ by A. By convention the empty context is omitted and parentheses associate to the left so that $\Gamma = (A_1, \cdots, A_n)$ is a generic context of length n. In this situation $(x_1, \cdots, x_n) : \Gamma$ introduces explicitly named variables to access the components of Γ. We have for $i = 1 \cdots n$

$$(x_1, \cdots, x_n) : \Gamma \vdash x_i : A_i$$

We may also introduce an indeterminate tuple of variables by $\gamma : \Gamma$. Then if $\Gamma = \underline{1}$ no variable is introduced and if $\Gamma = (\Gamma', A)$ then $\gamma.1$ is a tuple of variables for Γ' and $\gamma.2 : A$. For example if $\Gamma = (A_1, \ldots A_n)$ as before then

$$\gamma : \Gamma \vdash \gamma.1.1.2 : A_{n-2}$$

These two notations can be mixed as in

$$(\gamma, x) : (\Gamma, A) \vdash x : A$$

or even

$$\gamma : \Gamma,\ x : A \vdash x : A$$

If we want to emphasize free variables we may also write

$$\gamma : \Gamma \vdash A(\gamma)$$

instead of $\Gamma \vdash A$ and also write more explicitly $\underline{\Sigma}\gamma : \Gamma.A(\gamma)$ for the context extension (Γ, A). Note that $\underline{\Sigma}$ is a meta-operator on contexts and not part of the theory itself.

Similar conventions are adopted for tuples of terms. If $\Delta = (A_1, \cdots A_n)$ then we write

$$\Gamma \vdash (M_1, \cdots, M_n) : \Delta$$

if $\Gamma \vdash M_1 : A_1$, $\Gamma \vdash M_2 : A_2(M_1)$, etc. . We also use an indeterminate notation

$$\Gamma \vdash M : \Delta$$

where automatically $M = ()$ if $\Delta = \underline{1}$ and $M = (M.1, M.2)$ if $\Delta = \underline{\Sigma}\delta' : \Delta'.A(\delta')$ and then

$$\begin{array}{l} \Gamma \vdash M.1 : \Delta' \\ \Gamma \vdash M.2 : A(M.1) \end{array}$$

Such a tuple of terms $\Gamma \vdash M : \Delta$ is called a *context morphism* from Γ to Δ.

We single out specific tuples of types as so-called *telescopes*[1] or *contexts relative to a context*. We write $\Gamma \vdash \Delta$ if Δ is a context relative to Γ. We always have $\Gamma \vdash ()$ and if $\Gamma \vdash \Delta$ and $\Gamma, \Delta \vdash A$ then $\Gamma \vdash (\Delta, A)$. The above conventions on names, variables, and terms extend accordingly to telescopes. For example we write

$$x, y : \mathrm{N} \vdash \underline{\Sigma}(z, p) : (\underline{\Sigma} z : \mathrm{N}.\mathrm{Eq}\,(x, z)).\mathrm{Eq}\,(y, z)$$

as an abbreviation for the three judgements

$$\begin{aligned} &x, y : \mathrm{N} \vdash \mathrm{N} \\ &x, y : \mathrm{N},\ z : \mathrm{N} \vdash \mathrm{Eq}\,(x, z) \\ &x, y : \mathrm{N},\ z : \mathrm{N},\ p : \mathrm{Eq}\,(x, z) \vdash \mathrm{Eq}\,(y, z) \end{aligned}$$

and

$$x, y : \mathrm{N} \vdash M : \underline{\Sigma}(z, p) : (\underline{\Sigma} z : \mathrm{N}.\mathrm{Eq}\,(x, z)).\mathrm{Eq}\,(y, z)$$

as an abbreviation for

$$\begin{aligned} &x, y : \mathrm{N} \vdash M.1.1.2 : \mathrm{N} \\ &x, y : \mathrm{N} \vdash M.1.2 : \mathrm{Eq}\,(x, M.1.1.2) \\ &x, y : \mathrm{N} \vdash M.2 : \mathrm{Eq}\,(y, M.1.1.2) \end{aligned}$$

Remember that $M.1.1.1 = ()$.

2.3 Contexts of setoids

We are now ready to define the interpretation of contexts in the model. It may seem strange that it comes before the definition of (dependent) types, but as we shall see the contexts can be defined independently of the types and thus the inherent circularity in the definition of type dependency can be unravelled.

Definition 4. A *context of setoids* is a triple $\Gamma = (\Gamma_{\mathrm{set}}, \Gamma_{\mathrm{rel}}, \Gamma_{\mathrm{refl}})$ where Γ_{set} is a context and Γ_{rel} is a (syntactic) context relative to $(\Gamma_{\mathrm{set}}, \Gamma_{\mathrm{set}})$, i.e.

$$\gamma : \Gamma_{\mathrm{set}},\ \gamma' : \Gamma_{\mathrm{set}} \vdash \Gamma_{\mathrm{rel}}(\gamma, \gamma')$$

and finally Γ_{refl} is a tuple of terms proving reflexivity of Γ_{rel}, i.e.

$$\gamma : \Gamma_{\mathrm{set}} \vdash \Gamma_{\mathrm{refl}}(\gamma) : \Gamma_{\mathrm{rel}}(\gamma, \gamma)$$

Two contexts of setoids are equal if their three components are definitionally equal. The set of contexts of setoids is denoted Con.

[1] This term was introduced by N. De Bruijn.

Examples. If $\vdash A$ and $a, a' : A \vdash R(a, a')$ then we get a context of setoids Γ by putting

$$\begin{aligned} \Gamma_{\text{set}} &:= \underline{\Sigma} a : A.R(a, a) \\ \Gamma_{\text{rel}}((a,p),(a',p')) &:= (R(a,a')) \\ \Gamma_{\text{refl}}(a,p) &:= (p) \end{aligned}$$

So in particular every setoid gives rise to a context of setoids. As can be seen in this example the relation associated to a context of setoids is supposed to be reflexive because such a context is meant to contain a reflexivity assumption for every variable which is projected out by $-_{\text{refl}}$.

We could require the relation Γ_{rel} to be symmetric and transitive, too; but for reasons which become clear later in 3.1; these additional assumptions could never be used so they can be omitted.

Another example is the empty context of setoids $\bullet$ defined by $\bullet_{\text{set}} = \underline{1}$, $\bullet_{\text{rel}} = \underline{1}$, $\bullet_{\text{refl}} = ()$.

Next we define morphisms between contexts of setoids which are to interpret substitutions and weakenings.

Definition 5. Let Γ and Δ be contexts of setoids. A *morphism* from Γ to Δ is a pair $f = (f_{\text{fun}}, f_{\text{resp}})$ where

$$\gamma : \Gamma_{\text{set}} \vdash f_{\text{fun}}(\gamma) : \Delta_{\text{set}}$$

and

$$\gamma, \gamma' : \Gamma_{\text{set}},\ p : \Gamma_{\text{rel}}(\gamma, \gamma') \vdash f_{\text{resp}}(p) : \Delta_{\text{rel}}(f_{\text{fun}}(\gamma), f_{\text{fun}}(\gamma'))$$

such that reflexivity is preserved up to *definitional* equality, i.e.

$$\gamma : \Gamma \vdash f_{\text{resp}}(\Gamma_{\text{refl}}(\gamma)) = \Delta_{\text{refl}}(f_{\text{fun}}(\gamma))$$

is provable. Two morphisms between contexts of setoids are equal if their two components are definitionally equal. The set of these morphisms is denoted by $\text{Mor}\,(\Gamma, \Delta)$.

Examples. For every context of setoids Γ we have the *identity morphism* $\text{Id}_\Gamma \in \text{Mor}\,(\Gamma, \Gamma)$ given by

$$\begin{aligned} (\text{Id}_\Gamma)_{\text{fun}}(\gamma) &:= \gamma \\ (\text{Id}_\Gamma)_{\text{resp}}(p) &:= p \end{aligned}$$

Similarly, if $g \in \text{Mor}\,(\Delta, \Theta)$ and $f \in \text{Mor}\,(\Gamma, \Delta)$ we define the *composition* $g \circ f \in \text{Mor}\,(\Gamma, \Theta)$ by

$$\begin{aligned} (g \circ f)_{\text{fun}}(\gamma) &:= g_{\text{fun}}(f_{\text{fun}}(\gamma)) \\ (g \circ f)_{\text{resp}}(p) &:= g_{\text{resp}}(f_{\text{resp}}(p)) \end{aligned}$$

as componentwise substitution. Finally there is a unique element $!_\Gamma$ in Mor $(\Gamma, \bullet)$ defined by

$$(!_\Gamma)_{\mathrm{fun}}(\gamma) := ()$$
$$(!_\Gamma)_{\mathrm{resp}}(\gamma) := ()$$

Clearly in all three cases reflexivity is preserved so that morphisms have been defined, and thus we get the following proposition by straightforward calculation.

Proposition 6. *The contexts of setoids together with their morphisms form a category with terminal object* $\bullet$.

2.4 Families of setoids

The most important ingredient needed to interpret Martin-Löf type theory is the correct definition of type dependency. So we must say what a family of setoids indexed over a context of setoids is. We first introduce a useful abbreviation. If $\Gamma \in \mathrm{Con}$ and $\gamma_1, \ldots, \gamma_n : \Gamma_{\mathrm{set}}$ then

$$\Gamma_{\mathrm{conn}}(\gamma_1, \ldots, \gamma_n)$$

denotes the context consisting of all types $\Gamma_{\mathrm{rel}}(\gamma_i, \gamma_j)$ for $i, j = 1 \ldots n, i \neq j$. So for example if $\gamma, \gamma' : \Gamma_{\mathrm{set}}$ then

$$\Gamma_{\mathrm{conn}}(\gamma, \gamma') = (\Gamma_{\mathrm{rel}}(\gamma, \gamma'),\ \Gamma_{\mathrm{rel}}(\gamma', \gamma))$$

Definition 7. Let Γ be a context of setoids. A *family of setoids above* Γ is a 6-tuple $S = (S_{\mathrm{set}}, S_{\mathrm{rel}}, S_{\mathrm{rewrite}}, S_{\mathrm{sym}}, S_{\mathrm{trans}}, S_{\mathrm{ax}})$ where

- S_{set} is a type depending on Γ_{set}

$$\gamma : \Gamma_{\mathrm{set}} \vdash S_{\mathrm{set}}(\gamma)$$

- S_{rel} is a dependent relation on S_{set}

$$\gamma, \gamma' : \Gamma_{\mathrm{set}},\ s : S_{\mathrm{set}}(\gamma),\ s' : S_{\mathrm{set}}(\gamma') \vdash S_{\mathrm{rel}}(s, s')$$

- S_{rewrite} is a "rewriter" which allows to substitute connected elements of Γ_{set} in S_{set}

$$\gamma, \gamma' : \Gamma_{\mathrm{set}},\ p : \Gamma_{\mathrm{conn}}(\gamma, \gamma'),\ s : S_{\mathrm{set}}(\gamma),\ \bar{s} : S_{\mathrm{rel}}(s, s) \vdash S_{\mathrm{rewrite}}(p, s, \bar{s}) : S_{\mathrm{set}}(\gamma')$$

such that S_{rel} is symmetric and transitive and such that $S_{\text{rewrite}}(p, s)$ is always S_{rel}-related to s. More precisely we require terms

$$\begin{array}{ll}
\gamma, \gamma' : \Gamma_{\text{set}}, & \\
p : \Gamma_{\text{conn}}(\gamma, \gamma'), & \\
s : S_{\text{set}}(\gamma),\ s' : S(\gamma'), & \\
q : S_{\text{rel}}(s, s') & \vdash S_{\text{sym}}(p, q) : S_{\text{rel}}(s', s)
\end{array}$$

$$\begin{array}{ll}
\gamma, \gamma', \gamma'' : \Gamma_{\text{set}}, & \\
p : \Gamma_{\text{conn}}(\gamma, \gamma', \gamma''), & \\
s : S_{\text{set}}(\gamma),\ s' : S_{\text{set}}(\gamma'),\ s'' : S_{\text{set}}(\gamma''), & \\
q : S_{\text{rel}}(s, s'),\ q' : S_{\text{rel}}(s', s'') & \vdash S_{\text{trans}}(p, q, q') : S_{\text{rel}}(s, s'')
\end{array}$$

$$\begin{array}{l}
\gamma, \gamma' : \Gamma_{\text{set}}, \\
p : \Gamma_{\text{conn}}(\gamma, \gamma'), \\
s : S_{\text{set}}(\gamma),\ \bar{s} : S_{\text{rel}}(s, s) \vdash S_{\text{ax}}(p, s, \bar{s}) : S_{\text{rel}}(s, S_{\text{rewrite}}(p, s, \bar{s}))
\end{array}$$

Two families are equal if all their six components are definitionally equal. We denote the set of families of setoids over context Γ by $\text{Fam}(\Gamma)$.

The idea behind the rewriter "S_{rewrite}" is to allow substitution inside a dependent family if the indexing elements are "equal", i.e related. It will be the main ingredient in the interpretation of the identity elimination rule in the setoid model. The reflexivity assumption $\bar{s} : S_{\text{rel}}(s, s)$ in S_{ax} is needed for otherwise from S_{ax} one could conclude that any two elements of $S_{\text{set}}(\gamma)$ are S_{rel}-related. The rewriter S_{rewrite} could also be defined without this assumption.

Example. If Γ is a context of setoids and S is a setoid as defined in Section 2 then we can form a constant family of setoids indexed over Γ whose underlying set and relation are just their weakened companions taken from S, whereas the reindexer is the identity. In fact the nondependent setoids are in 1-1 correspondence to families of setoids over the empty context. More examples arise from the operations on families of setoids we are going to describe.

The task is now to show that this notion of families is stable under all constructions needed to interpret Martin-Löf type theory, i.e. to show that all six components of a family can be defined by induction along a type expression. This means that apart from the various type formers we are interested in, we must interpret context comprehension and substitution. This will be done in the rest of this section. A fully formal account of what exactly is needed in order to interpret Martin-Löf type theory can e.g. be found in [12].

2.5 Context comprehension

In the setoid model contexts are interpreted as contexts of setoids, whereas types (in contexts) will be interpreted as families of setoids above the interpretation of their context. So the first operation we have to define is context comprehension, i.e. the interpretation of the rule

$$\frac{\Gamma \vdash A}{\Gamma, x : A \text{ is a context}}$$

Let S be a family of setoids indexed over Γ. Its comprehension, denoted $\Gamma.S$, is the context of setoids defined by

$$\begin{array}{ll}
\Gamma.S_{\text{set}} & := \underline{\Sigma}(\gamma, s) : (\underline{\Sigma}\gamma \in \Gamma_{\text{set}}.S_{\text{set}}(\gamma)).S_{\text{rel}}(s, s) \\
\Gamma.S_{\text{rel}}((\gamma, s, p), (\gamma', s', p')) := & (\Gamma_{\text{rel}}(\gamma, \gamma'),\ S_{\text{rel}}(s, s')) \\
\Gamma.S_{\text{refl}}(\gamma, s, p) & := (\Gamma_{\text{refl}}(\gamma), p)
\end{array}$$

Thus a typical context of setoids Γ of length 2 takes the form

$$\begin{array}{ll}
\Gamma_{\text{set}} & = s_1 : (S_1)_{\text{set}}, \bar{s}_1 : (S_1)_{\text{rel}}(s_1, s_1), \\
 & s_2 : (S_2)_{\text{set}}(s_1, \bar{s}_1), \bar{s}_2 : (S_2)_{\text{rel}}(s_1, \bar{s}_1,\ s_2, s_2) \\
\Gamma_{\text{rel}}(s_1, \bar{s}_1, s_2, \bar{s}_2,\ s'_1, \bar{s}'_1, s'_2, \bar{s}'_2) & = p_1 : (S_1)_{\text{rel}}(s_1, s'_1), \\
 & p_2 : (S_2)_{\text{rel}}(s_1, \bar{s}_1, s'_1, \bar{s}'_1, s_2, s'_2) \\
\Gamma_{\text{refl}}(s_1, \bar{s}_1, s_2, \bar{s}_2) & = (\bar{s}_1, \bar{s}_2)
\end{array}$$

where we have used the standard notation for contexts to exemplify the use of the abbreviations.

Canonical projection. We define a morphism $\mathrm{p}(S) \in \mathrm{Mor}\,(\Gamma.S, \Gamma)$ by

$$\begin{array}{l}
\mathrm{p}(S)_{\text{fun}}(\gamma, s, \bar{s}) := \gamma \\
\mathrm{p}(S)_{\text{resp}}(p, q) \ \ := p
\end{array}$$

It follows immediately from the definition of $(\Gamma.S)_{\text{refl}}$ that this is indeed a morphism.

2.6 Sections of families

Instead of defining arbitrary morphisms between families we restrict ourselves to "sections" which will be used to interpret terms and can be viewed as family morphisms from the constant (unit) family corresponding to the context itself into a family.

Definition 8. If S is a family over a context of setoids Γ then a section of S is a pair $M = (M_{\text{el}}, M_{\text{resp}})$ where

$$\gamma : \Gamma_{\text{set}} \vdash M_{\text{el}}(\gamma) : S_{\text{set}}(\gamma)$$

and

$$\gamma, \gamma' : \Gamma_{\text{set}},\ p : \Gamma_{\text{rel}}(\gamma, \gamma') \vdash M_{\text{resp}}(p) : S_{\text{rel}}(M_{\text{el}}(\gamma), M_{\text{el}}(\gamma'))$$

We denote the set of sections of S by $\mathrm{Sect}\,(S)$. Two sections are equal if both components are definitionally equal.

Every section induces a context morphism from its context to the comprehension of its type. More precisely, if $M \in \mathrm{Sect}(S)$ then we get a morphism $\overline{M} \in \mathrm{Mor}(\Gamma, \Gamma.S)$ by

$$\begin{aligned}\overline{M}_{\mathrm{fun}}(\gamma) &:= (\gamma, M_{\mathrm{el}}(\gamma), M_{\mathrm{resp}}(\Gamma_{\mathrm{refl}}(\gamma)))\\ \overline{M}_{\mathrm{resp}}(p) &:= (p, M_{\mathrm{resp}}(p))\end{aligned}$$

We check that reflexivity is preserved:

$$\begin{aligned}&\overline{M}_{\mathrm{resp}}(\Gamma_{\mathrm{refl}}(\gamma)) =\\ &(\Gamma_{\mathrm{refl}}(\gamma),\ M_{\mathrm{resp}}(\Gamma_{\mathrm{refl}}(\gamma))) =\\ &\Gamma.S_{\mathrm{refl}}(\gamma, M_{\mathrm{el}}(\gamma), M_{\mathrm{resp}}(\Gamma_{\mathrm{refl}}(\gamma))) =\\ &\Gamma.S_{\mathrm{refl}}(\overline{M}_{\mathrm{fun}}(\gamma))\end{aligned}$$

So although no equational constraints are placed on sections they nevertheless induce proper morphisms.

We have

$$\mathrm{p}(S) \circ \overline{M} = \mathrm{Id}_{\Gamma}$$

which explains the term “section”. We can also get back sections from certain context morphisms as is shown in the next section.

2.7 Weakening and substitution

In a model, substitution and weakening are defined as additional operations rather than inductively as in the syntax. First, in this way substitution is defined for arbitrary families and sections, not only for the syntactically expressible ones. Second, with an explicit presentation of substitution it becomes easier to prove the correctness of the interpretation of the Π-type and other constructions which involve substitution. Of course, one must then show that the interpretation of syntactic substitution agrees with the semantical substitution applied to the interpretation of the original non-substituted syntactic object. For this, one must establish that all semantic type and term formers commute with semantic substitution.

Instead of defining the substitution of an element (a section) into a family we define substitution for arbitrary morphisms between contexts of setoids which gives both simultaneous substitution by several terms and weakening as special cases. This technique is reminiscent of categorical models of type theory. Under this interpretation it is helpful to think of morphisms of contexts as of tuples of terms (“context morphisms” or “substitutions”).

So let S be a family of setoids indexed over Δ and f be a morphism from Γ to Δ. We obtain a family of setoids over Γ denoted $S[f]$ by precomposing with f, i.e. by putting

$$S[f]_{\mathrm{set}}(\gamma : \Gamma_{\mathrm{set}}) := S_{\mathrm{set}}(f_{\mathrm{fun}}(\gamma))$$

$$S[f]_{\mathrm{rel}}(s : S_{\mathrm{set}}(f_{\mathrm{fun}}(\gamma)),\ s' : S_{\mathrm{set}}(f_{\mathrm{fun}}(\gamma'))) := S_{\mathrm{rel}}(s, s')$$

$$\begin{aligned}&S[f]_{\mathrm{rewrite}}(p : \Gamma_{\mathrm{conn}}(\gamma, \gamma'),\ s : S_{\mathrm{set}}(f_{\mathrm{fun}}(\gamma)),\ \bar{s} : S_{\mathrm{rel}}(s, s)) :=\\ &\quad S_{\mathrm{rewrite}}(f_{\mathrm{resp}}(p), s, \bar{s})\end{aligned}$$

Here by a slight abuse of notation we have applied f_{resp} to $p : \Gamma_{\text{conn}}(\gamma, \gamma')$. This is understood componentwise. The other components of $S[f]$ are defined similarly by precomposition.

Substitution along a morphism of the form $\overline{M}$ arising from a section corresponds to real substitution as in

$$\frac{\Gamma,\ x : S \vdash T(x)}{\Gamma \vdash T(M)}$$

whereas substitution along a morphism $\mathrm{p}(S)$ interprets weakening

$$\frac{\Gamma \vdash T}{\Gamma,\ x : S \vdash T}$$

If no confusion can arise we abbreviate $T[\mathrm{p}(S)]$ for $S, T \in \mathrm{Fam}(\Gamma)$ simply by T^+ and also write $T[M]$ instead of $T[\overline{M}]$ if $T \in \mathrm{Fam}(\Gamma.S)$ and $M \in \mathrm{Sect}\,(S)$.

There is also a context morphism arising from substitution which is a bit difficult to understand at first. It goes from $\Gamma.S[f]$ to $\Delta.S$ and we denote it by $q(f, S)$. Its function component is defined by

$$q(f,S)_{\text{fun}}(\gamma : \Gamma_{\text{set}},\ s : S_{\text{set}}(f_{\text{fun}}(\gamma)),\ \bar{s} : S_{\text{rel}}(s,s)) := (f_{\text{fun}}(\gamma), s, \bar{s}) : \Delta.S_{\text{set}}$$

It is used to perform substitutions in variables other than the last one as in

$$\frac{\begin{array}{l}\Gamma \vdash A\\ \Gamma,\ x : A \vdash B\\ \Gamma,\ x : A,\ y : B \vdash C\\ \Gamma \vdash M : A\end{array}}{\Gamma,\ y : B(M) \vdash C(M,y)}$$

In the model, A is a family of setoids over Γ, B is one over $\Gamma.A$, and C is a family over $\Gamma.A.B$. M is a section of A. The conclusion is then obtained as

$$C[q(M,B)]$$

It is a characteristic property of substitution that the square of morphisms f, $\mathrm{p}(S)$, $\mathrm{p}(S[f])$, $q(f,S)$ is a pullback.

Extraction of sections from morphisms. If $f \in \mathrm{Mor}\,(\Gamma, \Delta.S)$ then we construct a section $\mathrm{Fst}(f) \in \mathrm{Sect}\,(S[\mathrm{p}(S) \circ f])$ as follows. The $-_{\text{el}}$-part of $\mathrm{p}(S) \circ f$ sends $\gamma : \Gamma_{\text{set}}$ to $\delta := f_{\text{fun}}(\gamma).1.1.2 : \Delta_{\text{set}}$. Now $f_{\text{fun}}(\gamma).1.2$ is an element of $S_{\text{set}}(\delta)$. So we put

$$\mathrm{Fst}(f)_{\text{el}}(\gamma) := f_{\text{fun}}(\gamma).1.2$$

The $-_{\text{resp}}$-component is defined analogously. We have

$$f = q(f', S) \circ \overline{\mathrm{Fst}(f)}$$

and
$$M = \mathrm{Fst}(\overline{M})$$
so sections and right-inverses to canonical projections are in 1-1 correspondence. If $f \in \mathrm{Mor}(\Gamma, \Delta)$, $S \in \mathrm{Fam}(\Delta)$ and $M \in \mathrm{Sect}(S[f])$ then
$$\mathrm{Cons}(f, M) := q(f, S) \circ \overline{M} \in \mathrm{Mor}(\Gamma, \Delta.S)$$
and $\mathrm{Fst}(\mathrm{Cons}(f, M)) = M$. This may explain the operator name Fst.

Substitution and weakening on sections. Let $f \in \mathrm{Mor}(\Gamma, \Delta)$ and $S \in \mathrm{Fam}(\Delta)$. If M is a section of S then
$$M[f] := \mathrm{Fst}(\overline{M} \circ f)$$
is a section of $S[f]$. Its element component is
$$M[f]_{\mathrm{el}}(\gamma : \Gamma_{\mathrm{set}}) = M_{\mathrm{el}}(f_{\mathrm{fun}}(\gamma))$$
The $-_{\mathrm{resp}}$-component is defined similarly. In this way we interpret substitution on terms. The abbreviations introduced before apply analogously to substitution on sections.

Variables. The final ingredient we require to interpret all manoeuvres with variables and substitutions is the extraction of the last variable in a context, i.e. the interpretation of
$$\Gamma,\ x : S \vdash x : S$$
In our setup this is a section of S^{+}, since the type S on the rhs is actually weakened. We obtain this section as
$$\mathrm{var}^{S} := \mathrm{Fst}(\mathrm{Id}_{\Gamma.S})$$
Its first component is
$$\mathrm{var}^{S}{}_{\mathrm{el}}(\gamma : \Gamma_{\mathrm{set}}, s : S_{\mathrm{set}}(\gamma), \bar{s} : S_{\mathrm{rel}}(s, s)) = s$$
as expected. The other variables are obtained as suitable weakenings of this.

It should be stressed that our substitution inherits the split property from the syntax. This means that for $f \in \mathrm{Mor}(\Gamma, \Delta)$ and $g \in \mathrm{Mor}(\Delta, \Theta)$ and S a family above Θ we have the equality of families
$$S[g][f] = S[g \circ f]$$
and also
$$S[\mathrm{Id}_{\Theta}] = S$$
The structure of the model is now set out, we have defined domains of interpretation for contexts, types, and terms and related them through appropriate operations. It remains to show that the model contains the base types and is closed under the type constructors we are interested in. Since Martin-Löf's type theory is an open system there can be arbitrarily many such constructors. For the purpose of this paper we confine ourselves to the interpretation of the dependent product, and the intensional identity type (together with Ext). In Section 5 we sketch the interpretation of other type constructors like inductive types and universes.

3 Dependent product

Let Γ be a context of setoids, S be a family over Γ and T be a family over $\Gamma.S$. We want to define a family over Γ which "internalises" the sections of T, in order to interpret the Π-type. We denote this family by $\Pi(S,T)$. Its type component is given by

$$\Pi(S,T)_{\text{set}}(\gamma : \Gamma_{\text{set}}) := \Pi s : S_{\text{set}}(\gamma).\Pi \bar{s} : S_{\text{rel}}(s,s).T_{\text{set}}(\gamma, s, \bar{s})$$

The relation is defined by

$$\begin{aligned}&\Pi(S,T)_{\text{rel}}(U : \Pi(S,T)_{\text{set}}(\gamma),\ V : \Pi(S,T)_{\text{set}}(\gamma')) := \\ &\quad \Pi s : S_{\text{set}}(\gamma)\Pi \bar{s} : S_{\text{rel}}(s,s)\Pi s' : S_{\text{set}}(\gamma')\Pi \bar{s}' : S_{\text{rel}}(s',s'). \\ &\qquad S_{\text{rel}}(s,s') \to T_{\text{rel}}(U\,s\,\bar{s}, V\,s'\,\bar{s}')\end{aligned}$$

We leave out the definition of the other components, but mention that the proof of transitivity requires the component S_{rewrite}. This means that if we had defined families of setoids without the rewriter, it would have been impossible to define the dependent product. We also need the $-_{\text{refl}}$ operation in the definition of various parts.

Next we define introduction and elimination for the dependent product, that is the interpretation of the two rules:

$$\frac{\Gamma, x : S \vdash M(x) : T(x)}{\Gamma \vdash \lambda x{:}S.M(x) : \Pi x{:}S.T(x)} \qquad \frac{\Gamma \vdash M : \Pi x{:}S.T(x) \quad \Gamma \vdash N : S}{\Gamma \vdash (M\,N) : T(N)}$$

So if $M \in \text{Sect}\,(T)$ we construct

$$\Pi_{\text{intro}}(S,T,M) \in \text{Sect}\,(\Pi(S,T))$$

and conversely if $M \in \text{Sect}\,(\Pi(S,T))$ and $N \in \text{Sect}\,(S)$ we construct

$$\Pi_{\text{elim}}\,(S,T,M,N) \in \text{Sect}\,(T[N])$$

The element parts are

$$\Pi_{\text{intro}}(S,T,M)_{\text{el}}(\gamma : \Gamma_{\text{set}}) := \lambda s : S_{\text{set}}(\gamma).\lambda \bar{s} : S_{\text{rel}}(s,s).M_{\text{el}}(\gamma, s, \bar{s})$$

and

$$\Pi_{\text{elim}}\,(S,T,M,N)_{\text{el}}(\gamma : \Gamma_{\text{set}}) := M_{\text{el}}(\gamma)\;N_{\text{el}}(\gamma)\;N_{\text{resp}}(\Gamma_{\text{refl}}(\gamma))$$

The relational parts are defined similarly. Now for $M \in \text{Sect}\,(T)$ and $N \in \text{Sect}\,(S)$ we have the equation

$$\Pi_{\text{elim}}\,(S,T,\Pi_{\text{intro}}\,(S,T,M),N) = M[N]$$

which interprets the β-equality rule. If the underlying type theory has an η-rule then a corresponding equation holds as well.

3.1 Compatibility with substitution.

As argued in 2.7 we must now show that substitution commutes with product formation, as well as with introduction and elimination. In categorical jargon this is referred to as the "Beck-Chevalley condition" or "stability under substitution". Consider the following situation: $\Gamma \in \mathrm{Con}$, S above Γ, and T above $\Gamma.S$ as before; Γ' another context and f a context morphism from Γ' to Γ. Now we may form the product of T and then substitute along f:

$$\Pi(S,T)[f]$$

or first substitute f into both S and T and then take the product:

$$\Pi(S[f],\ T[q(f,S)])$$

It can be shown by careful examination or by machine-supported normalisation that these families are indeed equal. It is crucial for this result that all morphisms preserve reflexivity ($-_{\mathrm{refl}}$), since this operation is part of the definition of the dependent product.

To clarify the delicacy of stability under substitution in the setoid model we give two type formers A and B which are not stable under substitution. Both simply map families over some context Γ to families over Γ. We define for $S \in \mathrm{Fam}(\Gamma)$

$$A(S)_{\mathrm{set}}(\gamma : \Gamma_{\mathrm{set}}) := \Pi\gamma' : \Gamma_{\mathrm{set}}.\Gamma_{\mathrm{conn}}(\gamma,\gamma') \to S_{\mathrm{set}}(\gamma')$$

By S_{rewrite} the family $A(S)$ will be "isomorphic" to S up to S_{rel}, but no matter how the other components of $A(S)$ are defined, the type constructor $A(-)$ cannot be stable under substitution because for $f \in \mathrm{Mor}(\Gamma,\Delta)$ and $S \in \mathrm{Fam}(\Delta)$ we have

$$A(S)[f]_{\mathrm{set}}(\gamma) = \Pi\delta' \in \Delta_{\mathrm{set}}.\Delta_{\mathrm{conn}}(f_{\mathrm{fun}}(\gamma),\delta') \to S_{\mathrm{set}}(\delta')$$

whereas

$$A(S[f])_{\mathrm{set}}(\gamma) = \Pi\gamma' \in \Gamma_{\mathrm{set}}.\Gamma_{\mathrm{conn}}(\gamma,\gamma') \to S_{\mathrm{set}}(f_{\mathrm{fun}}(\gamma'))$$

For the other example we assume an operation $\Gamma_{\mathrm{sym}} : \Pi\gamma,\gamma' : \Gamma_{\mathrm{set}}.\Gamma_{\mathrm{rel}}(\gamma,\gamma') \to \Gamma_{\mathrm{rel}}(\gamma',\gamma)$ to be defined for each context Γ. This is in fact possible for all contexts defined from $\bullet$ by successive applications of comprehension. Now if for $S \in \mathrm{Fam}(\Gamma)$ we define

$$\begin{aligned}
B(S)_{\mathrm{set}} &:= S_{\mathrm{set}}\\
B(S)_{\mathrm{rel}} &:= S_{\mathrm{rel}}\\
B(S)_{\mathrm{sym}}((p_1,p_2):\Gamma_{\mathrm{conn}}(\gamma,\gamma'),\ q:S_{\mathrm{rel}}(s,s')) &:= S_{\mathrm{sym}}((\Gamma_{\mathrm{sym}}(p_2),\Gamma_{\mathrm{sym}}(p_1)),\ q)
\end{aligned}$$

i.e. we use Γ_{sym} to "permute" the proof (p_1,p_2), then again we lose stability under substitution because for $f \in \mathrm{Mor}(\Gamma,\Delta)$ and $S \in \mathrm{Fam}(\Delta)$ the family $B(S)[f]$ contains the term

$$\Delta_{\mathrm{sym}}(f_{\mathrm{resp}}(p_2))$$

whereas $B(S[f])$ contains

$$f_{\text{resp}}(\Gamma_{\text{sym}}(p_2))$$

and these are in general not equal. This is the reason why (in contrast to an earlier version of this paper) we do not require the relations in a context to be symmetric and transitive and why we use Γ_{conn} instead of Γ_{rel} as a premise to all operations associated with a family. We could of course require that context morphisms have to preserve symmetry and transitivity, but then also sections would have to do so and it would no longer be possible to internalise them in a Π-type.

In both examples we have used "local components" of Γ. In contrast, all type- and term operators the definition of which uses only Γ_{refl} and the components of the participating families will be stable under substitution.

4 The identity type

We can now reap the fruits of the laborious model constructions carried out above and define the identity setoid which satisfies the extensionality principle. The basic idea is that the identity setoid will be given by the relation associated to each family and the identity elimination operator will be based on the "rewriter" S_{rewrite}.

Suppose we are given a context of setoids Γ and a family of setoids S over Γ. We first form the context consisting of Γ and two copies of S. In our notation this is $\Gamma.S.S^+$. The identity setoid, denoted $\text{Eq}\,(S)$, is a family over this. There is a rather simple definition of this family if we are prepared to accept that in the model the equality rule Eq-Comp-J for the identity type holds only up to propositional equality, i.e. the interpretations of

$$\text{J}_C(M)(x, x, \text{refl}_A(x))$$

and

$$M(x)$$

are not equal, but the interpretation of the corresponding identity type is inhabited in the model. If we want the equality rule to hold in the model definitionally we must do a bit more work. In this paper we describe the first (simplified) identity family in detail because it explains the idea and only hint in 4.3 at the construction of the correct identity family since it is rather complicated and best understood using a proof checker like Lego interactively.

4.1 The simplified case

In the simplified case we put

$$\text{Eq}\,(S)_{\text{set}}((\gamma, s_1, \bar{s}_1, s_2, \bar{s}_2) : (\Gamma.S.S^+)_{\text{set}}) := S_{\text{rel}}(s_1, s_2)$$

and

$$\text{Eq}\,(S)_{\text{rel}}(p, q) = \mathbf{1}$$

i.e. any two elements of $\mathrm{Eq}\,(S)_{\mathrm{set}}$ are related. Clearly this is symmetric and transitive. To define the rewriter $\mathrm{Eq}\,(S)_{\mathrm{rewrite}}$ we make use of transitivity and symmetry of S_{rel}. We shall now embark on the definition of the combinators associated with the identity type.

Reflexivity. In the model $\mathrm{refl}_S(x)$ is a section of $\mathrm{Eq}\,(S)[\mathrm{var}^S]^2$. It is defined by

$$\mathrm{refl}\,(S)_{\mathrm{el}}((\gamma, s, \bar{s}) : \Gamma.S) := \bar{s}$$

The $-_{\mathrm{resp}}$ component is trivial since any two elements of the identity setoid are related.

Identity elimination. The main ingredient of the definition of the J-eliminator for $\mathrm{Eq}\,(S)$ is the rewriter $-_{\mathrm{rewrite}}$. We must do a bit of work, though, in order to get the various contexts and substitutions right. Let C be a family of setoids indexed over $\Gamma.S.S^+.\mathrm{Eq}\,(S)$. Substituting $\mathrm{refl}\,(S)$ into C gives

$$C(\mathrm{refl}\,(S)) := C[q(\mathrm{var}^S, \mathrm{Eq}\,(S))][\mathrm{refl}\,(S)]$$

Let M be a section of this family. We must construct a section $\mathrm{J}(C, M)$ of C from this. We define

$$\begin{aligned}
&\mathrm{J}(C, M)_{\mathrm{el}}((\gamma, s_1, \bar{s}_1, s_2, \bar{s}_2, p, \bar{p}) : \Gamma.S.S^+.\mathrm{Eq}\,(S)) := \\
&\quad C_{\mathrm{rewrite}}(\\
&\qquad ((\Gamma_{\mathrm{refl}}(\gamma),\ \bar{s}_1,\ p,\ \star),\ (\Gamma_{\mathrm{refl}}(\gamma),\ \bar{s}_1,\ S_{\mathrm{sym}}((\Gamma_{\mathrm{refl}}(\gamma), \Gamma_{\mathrm{refl}}(\gamma)), p),\ \star)), \\
&\qquad M_{\mathrm{el}}(\gamma, s_1, \bar{s}_1), \\
&\qquad M_{\mathrm{resp}}(\Gamma_{\mathrm{refl}}, \bar{s}_1))
\end{aligned}$$

Here $\star$ denotes the unique element of the unit type $\mathbf{1}$. The first argument to C_{rewrite} is a proof that

$$x := (\gamma, s_1, \bar{s}_1, s_1, \bar{s}_1, \bar{s}_1, \star)$$

and

$$y := (\gamma, s_1, \bar{s}_1, s_2, \bar{s}_2, p, \star)$$

are connected elements of $(\Gamma.S.S^+.\mathrm{Eq}\,(S))_{\mathrm{set}}$. The assumption M gives an element of $C_{\mathrm{set}}(x)$ (second argument) and a proof that it is related to itself (third argument). C_{rewrite} then gives the desired element of $C_{\mathrm{set}}(y)$.

We omit the $-_{\mathrm{resp}}$ part of $\mathrm{J}(C, M)$. Its main ingredient is C_{ax} — the proof that rewriting does not alter the C_{rel}-class. The definition of the other eliminator K is analogous.

As announced above this eliminator does not satisfy the rule EQ-COMP-J, i.e. we do not have

$$\mathrm{J}(C, M)[q(\mathrm{var}^S, \mathrm{Eq}\,(S))][\mathrm{refl}\,(S)] = M$$

The reason for this is that C_{rewrite} applied to a proof by reflexivity is not necessarily the identity function. However, by virtue of C_{ax}, the propositional companion to EQ-COMP-J, is inhabited in the model.

[2] Notice that $\mathrm{var}^S \in \mathrm{Mor}\,(\Gamma.S, (\Gamma.S.S^+)_{\mathrm{set}})$ is the "diagonal".

4.2 Extensionality

Let Γ and S be as before, T above $\Gamma.S$ and U, V sections of T. Moreover, assume a section M of

$$\mathrm{Eq}\,(S)(U,V) := \mathrm{Eq}\,(T)[V^+][U]$$

i.e. a proof that U and V are equal. We must show that their respective abstractions are equal as well — we need a section Ext (M) of

$$\mathrm{Eq}\,(\Pi(S,T))[\Pi_{\mathrm{intro}}(T,V)^+][\Pi_{\mathrm{intro}}(T,U)]$$

The element part of M is basically (modulo some currying of Σ-types) a term of type

$$\gamma : \Gamma_{\mathrm{set}},\ s : S_{\mathrm{set}}(\gamma),\ \bar{s} : S_{\mathrm{rel}}(s,s) \vdash S_{\mathrm{rel}}(U_{\mathrm{el}}(\gamma,s,\bar{s}),\ V_{\mathrm{el}}(\gamma,s,\bar{s}))$$

The element we are looking for amounts to an inhabitant of

$$\begin{array}{l}\gamma : \Gamma_{\mathrm{set}},\ s : S_{\mathrm{set}}(\gamma),\ \bar{s} : S_{\mathrm{rel}}(s,s),\ s' : S_{\mathrm{set}}(\gamma),\ \bar{s}' : S_{\mathrm{rel}}(s,s) \vdash \\ \quad S_{\mathrm{rel}}(U_{\mathrm{el}}(g,s,\bar{s}),\ V_{\mathrm{el}}(g,s',\bar{s}'))\end{array}$$

We obtain that by using either U_{resp} or V_{resp} and transitivity. The $-_{\mathrm{resp}}$ part is again trivial.

4.3 The unsimplified case

For the rule EQ-COMP-J to hold definitionally in the model we need a sum type in the original type theory given by the following rules.

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A + B} \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash \mathrm{inl}(M) : A + B} \quad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathrm{inr}(M) : A + B}$$

$$\frac{\begin{array}{l}\Gamma,\ x : A + B \vdash C(x) \\ \Gamma,\ x : A \vdash L(x) : C(\mathrm{inl}(x)) \\ \Gamma,\ x : B \vdash R(x) : C(\mathrm{inr}(x)) \\ \Gamma \vdash M : A + B\end{array}}{\Gamma \vdash \mathrm{cases}(L,R,M) : C(M)} \qquad \frac{}{\Gamma \vdash \mathrm{cases}(L,R,\mathrm{inl}(M)) = L(M) : C(\mathrm{inl}(M))} \quad \frac{}{\Gamma \vdash \mathrm{cases}(L,R,\mathrm{inr}(M)) = R(M) : C(\mathrm{inr}(M))}$$

With this sum type we can treat reflexivity as a special case and then interpret J as the identity. More precisely we put

$$\begin{array}{l}\mathrm{Eq}\,(S)_{\mathrm{set}}((\gamma,s_1,\bar{s}_1,s_2,\bar{s}_2) : (\Gamma.S.S^+)_{\mathrm{set}}) := \\ \quad (\Sigma p : \mathrm{Eq}(s_1,s_2).\mathrm{Eq}(\mathrm{Subst}\,(p,\bar{s}_1),\ \bar{s}_2))\ +\ S(s_1,s_2)\end{array}$$

So an element of the identity family either proves that the two arguments are equal or that they are related. The relation on the identity family is chosen to be trivial. The rewriting component is again defined using symmetry and transitivity. A rewritten element of the identity family always is in the right component of the sum.

For reflexivity we choose the left component.

$$\mathrm{refl}\,(S)_{\mathrm{el}}((\gamma, s, \bar{s}) : \Gamma.S) := \mathrm{inl}(\mathrm{refl}(s), \mathrm{refl}(\bar{s}))$$

Now the elimination rule J can be defined by case distinction. If the element $p : \mathrm{Eq}\,(S)(\gamma, s_1, \bar{s}_1, s_2, \bar{s}_2)$ is of the form $\mathrm{inl}((a, b))$ then we use the J-operator from the original type theory, whereas in the other case we use C_{rewrite} as in the other definition above. Now the equation EQ-COMP-J can be deduced from the equations for the sum type and the EQ-COMP-J-equation from the original type theory.

Similarly one may define K and Ext.

5 Interpretation of other type constructors

In this section we sketch the interpretation of type constructors other than Π and Eq. Some interpretations, in particular the one of the Σ-type below, only approximate their syntactical companions. Often, a problem similar to the one encountered with the simplified definition of the identity setoid occurs: a certain equality rule only holds propositionally. So in the interpretation one would have to replace all instances of the β-rule for equality by propositional equalities and suitably interspersed Subst-operations.

Inductive types like the natural numbers or the booleans lift straightforwardly to the setoid model; the relation is simply the identity type. The computation rule for the natural numbers will only be valid for closed terms and otherwise hold propositionally.

Parametrised inductive types like lists or trees remain to be investigated.

Universes. If the underlying type theory has a universe

$$\Gamma \vdash U$$

$$\Gamma,\ x : U \vdash \mathrm{El}(x)$$

then in the setoid model a universe can be defined as the discrete setoid (∇) on the type

$$\begin{array}{lr}
\Sigma S : U.\Sigma S_r : \mathrm{El}(S) \to \mathrm{El}(S) \to U. & \text{type and relation} \\
\quad (\Pi s, s' : \mathrm{El}(S).\mathrm{El}(S_r\ s\ s') \to \mathrm{El}(S_r\ s'\ s)) \ \times & \text{symmetry} \\
\quad (\Pi s, s', s'' : \mathrm{El}(S).\mathrm{El}(S_r\ s\ s') \to \mathrm{El}(S_r\ s'\ s'') \to \mathrm{El}(S_r\ s\ s'')) & \text{transitivity}
\end{array}$$

This universe is then closed under Π, Eq, and Σ, provided the original one (U) is. So in particular we can obtain a model of the Calculus of Constructions, i.e. Martin-Löf type theory with a universe closed under impredicative quantification.

Quotient types. The relation on a family of setoids being arbitrary we can e.g. interpret a type of rationals with underlying set N × N and suitable relation. More generally it is possible to interpret in the model a *quotient type former* which allows to form the quotient of a type by an arbitrary internal equivalence relation. This will be spelled out in more detail in the author's forthcoming thesis [3].

Streams. Among the numerous applications of the propositional ξ-rule we just single out the following. In the presence of Ext the type $\mathrm{N} \rightarrow A$ may serve as a type of streams (infinite lists) over A which has the property that any two bisimilar streams are propositionally equal and thus can be substituted for one another in every context.

Sigma types. Our definition of contexts and context morphisms is based on an equational constraint (preservation of reflexivity). So we cannot expect that they can easily be internalised in the setoid model by some sort of Σ-type. The obvious guess for a Σ-type in the setoid model has (for Γ, S, T as in the definition of the Π-type) as underlying set

$$\Sigma(S,T)_{\mathrm{set}}(\gamma : \Gamma_{\mathrm{set}}) = \Sigma s : S_{\mathrm{set}}(\gamma).\Sigma \bar{s} : S_{\mathrm{rel}}(s,s).T_{\mathrm{set}}(\gamma, s, \bar{s})$$

and as relation

$$\Sigma(S,T)_{\mathrm{rel}}((s,\bar{s},t),(s',\bar{s}',t')) = S_{\mathrm{rel}}(s,s') \times T_{\mathrm{rel}}(t,t')$$

We can now define pairing and the first projection and more generally interpret the weak Σ-elimination rule [10]. In order to define a second projection or equivalently the dependent Σ-elimination rule we must use T_{rewrite} and thus we cannot get the equation

$$(M,N).2 = N$$

but only interpret the corresponding propositional identity. If we insist on a full Σ-type we could move to telescopes of setoids, i.e. a family of setoids would be a list of the form $(\Gamma, S_1, \ldots, S_n)$ where the S_i are families in the old sense and S_1 is above Γ, S_2 is above $\Gamma.S_1$ and so forth. For general reasons this structure is a model, and has strong Σ-types.

6 Summary and conclusive remarks

We have now developed a full model of intensional Martin-Löf type theory. This means that along the lines described in [12] we can define a semantic function $[\![-]\!]$ which maps ordinary contexts to contexts of setoids, types in contexts to families of setoids over the interpretation of their contexts and finally terms to sections. This interpretation is sound in the sense that whenever some judgement J is derivable in the theory with Ext, then $[\![J]\!]$ is derivable in the pure theory by mimicking the derivation of J in the setoid model. The converse does not hold, i.e. if $[\![J]\!]$ is derivable then J itself need not be derivable in the theory with Ext. Consider for example the judgement J

$$\vdash \mathrm{refl}_{\mathrm{N}}(0) : \mathrm{Eq}_{\mathrm{N}}(0,\ \mathrm{Subst}\,(\mathrm{Ext}\,(P), 0))$$

where P is a proof that two functions on the natural numbers are pointwise propositionally equal. Clearly $[\![J]\!]$ is provable because Subst (Ext (P), 0) equals 0 in the setoid model. Intuitively this is so because $[\![\mathrm{Subst}\,(\mathrm{Ext}\,(p), 0)]\!]$ is a term of type $[\![\mathrm{N}]\!] = \mathrm{N}$ in the empty context. So it must be a normal form of type N, but it can hardly be a successor. On the other hand J itself is not derivable as indicated in the Introduction.

So a precise "specification" of the setoid model is not straightforward. For the moment we can only offer the semantic equations themselves. In future work we would like to establish a correspondence between the setoid model and derivations in extensional type theory, i.e. with the equality reflection rule.

We have argued that we can now use the type theory with Ext and eliminate all occurrences of Ext by a post-hoc interpretation in the setoid model. But is there any use in actually performing this interpretation? Certainly the terms obtained from this interpretation can be executed, whereas the terms containing Ext cannot. But if one is merely interested in execution one may just as well use the much simpler realisability interpretation into untyped lambda calculus where Ext can be realised by the identity.

In our opinion the real value of model constructions like the one we have described lies in the syntactic justification of rules like Ext or quotient types, and in the insight that formal proof development in type theory does not necessarily mean to carry around lots of different equalities and substitutivity proofs, since in principle the translation into setoids may always be performed.

There are at least two more solutions to the extensionality problem. One consists of adding a new universe of propositions which does not affect the types at all. One is then free to add any propositional assumptions one likes, provided one can give a model or other proof of consistency, but one loses the possibility of computational use of proofs, in particular the possibility to reindex dependent types along propositional equalities. If one is interested in formalisations of algebra where dependent types play a subordinate rôle, this is a very simple yet sound approach.

The other solution is again a syntactic model construction, which differs from the given one in that we interpret families of setoids as nondependent sets where a non-reflexive relation singles out the different fibres. The identity type becomes the unit type under this interpretation. In this model the type part and the relation part are completely separated, so that the relation part can be projected away. What remains is a kind of realisability interpretation of type theory. Dependent types are interpreted as simple types, dependent product as the arrow type and the identity type becomes the unit. Both constructions will be described in more detail in [3], too.

The setoid model has been implemented in the Lego system [8]. This means that the combinators we have defined are actually available and setoids can be built using these combinators according to some derivation in type theory. Also we have used Lego to check all the equations which are required to hold. For the future it might be useful to have a "parser" which translates actual lambda terms into such combinators.

Acknowledgements

I wish to thank Thorsten Altenkirch, Don Sannella, Thomas Streicher, and an anonymous referee for helpful comments on earlier versions of this paper.

References

1. J. Cartmell. *Generalized algebraic theories and contextual categories.* PhD thesis, Univ. Oxford, 1978.
2. Solomon Feferman. Theories of Finite Type. In Jon Barwise, editor, *Handbook of Mathematical Logic*, chapter D.4, pages 934–935. North-Holland, 1977.
3. Martin Hofmann. *Extensional concepts in intensional type theory.* PhD thesis, Univ. of Edinburgh, forthcoming 1994.
4. Bart Jacobs. *Categorical Type Theory.* PhD thesis, University of Utrecht, 1991.
5. Bart Jacobs. Comprehension categories and the semantics of type theory. *Theoretical Computer Science*, 107:169–207, 1993.
6. Horst Luckhardt. *Extensional Gödel Functional Interpretation. A Consistency Proof of Classical Analysis*, volume 306 of *Lecture Notes in Mathematics.* Springer, Berlin, 1973.
7. Zhaohui Luo. *An Extended Calculus of Constructions.* PhD thesis, University of Edinburgh, July 1990.
8. Zhaohui Luo and Randy Pollack. LEGO Proof Development System: User's Manual. Technical Report ECS-LFCS-92-211, University of Edinburgh, May 1992.
9. Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis·Napoli, 1984.
10. John Mitchell and Gordon Plotkin. Abstract Types have Existential Type. *ACM Transactions on Programming Languages and Systems*, 10(3):470–502, 1988.
11. B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löfs Type Theory, An Introduction.* Clarendon Press, Oxford, 1990
12. Thomas Streicher. *Semantics of Type Theory.* Birkhäuser, 1991.
13. Thomas Streicher. *Semantical Investigations into Intensional Type Theory.* Habilitationsschrift, LMU München, to appear 1993.
14. David Turner. A new formulation of constructive type theory. In P. Dybjer, editor, *Proceedings of the Workshop on Programming Logic*, pages 258–294. Programming Methodology Group, Univ. of Göteborg, May 1989.