Conservativity of Equality Reflection over Intensional Type Theory

Martin Hofmann

Fachbereich Mathematik
TH Darmstadt
Schloßgartenstraße 7
D-64289 Darmstadt
Germany

Abstract. We investigate the relationship between intensional and extensional formulations of Martin-Löf type theory. We exhibit two principles which are not provable in the intensional formulation: uniqueness of identity and functional extensionality. We show that extensional type theory is conservative over the intensional one extended by these two principles, meaning that the same types are inhabited, whenever they make sense. The proof is non-constructive because it uses set-theoretic quotienting and choice of representatives.

1 Extensional and intensional type theory

A distinctive feature of Martin-Löf's type theories is the presence of two notions of equality: judgemental equality and propositional equality. Judgemental equality applies to both terms and types and is written as a judgement $\Gamma \vdash M = N : \sigma$ and $\Gamma \vdash \sigma = \tau$ *type*. The inference rules for these judgements include congruence rules for all the type and term formers and computational rules like the β and possibly the η-rule for function types. The present study applies to dependent type theories with at least Π-types, Σ-types, identity types, and natural numbers (to have a base type). For simplicity we assume an η-rule for the Π-types. In [6] we also deal with the absence of the η-rule.

Propositional equality only applies to terms and is itself a type (of equality proofs). That is, we have a type $Id_\sigma(M, N)$ in context Γ whenever $M, N : \sigma$.

$$\frac{\Gamma \vdash \sigma \ \textit{type} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash Id_\sigma(M, N) \ \textit{type}} \ \textsc{Id-Form}$$

The formulation of equality as a type allows internal reasoning about equality, for example a propositional equality can be established by induction (**N**-elimination) so as to show $x{:}\mathbf{N} \vdash Id_{\mathbf{N}}(x, Suc^x(0))$ *true* where $Suc^x(0)$ means the x-fold application of the successor function to 0 defined by **N**-elimination. Here and in the sequel we write $\Gamma \vdash \sigma$ *true* to mean $\Gamma \vdash \sigma$ *type* and $\Gamma \vdash M : \sigma$ for some term M. Type annotations to *Id* and other type and term formers may be left out where appropriate.

Propositional equality is introduced via reflexivity, i.e. if $M : \sigma$ then we have a canonical element $Refl_\sigma(M) : Id_\sigma(M, M)$.

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash Refl_\sigma(M) : Id_\sigma(M, M)} \quad \text{ID-INTRO}$$

This, together with the congruence rules for judgemental equality gives that $M = N : \sigma$ always entails that $Id_\sigma(M, N)$ is inhabited (namely by $Refl_\sigma(M)$).

Now one would like to express that this is the only inhabitant of the identity type, i.e. one might wish the rule

$$\frac{\Gamma \vdash P : Id_\sigma(M, N)}{\Gamma \vdash P = Refl_\sigma(M) : Id_\sigma(M, N)} \quad \text{ID-UNI}$$

However, for this rule to "type-check" we need to identify propositional and judgemental equality. This is achieved by the equality reflection rule

$$\frac{\Gamma \vdash P : Id_\sigma(M, N)}{\Gamma \vdash M = N : \sigma} \quad \text{REFLECTION}$$

This axiomatisation of propositional equality is the one used in Martin-Löf's earlier works [8] and in many proof-theoretic studies on his type theories. The type theory based on this axiomatisation is called *extensional type theory*, TT_E for short, because by rule REFLECTION the judgemental equality becomes extensional. For instance, since as argued above $x{:}\mathrm{N} \vdash Id_\mathrm{N}(x, Suc^x(0))$ *true* we obtain by REFLECTION that $x{:}\mathrm{N} \vdash x = Suc^x(0) : \mathrm{N}$ which clearly is an extensional equality.

Equality reflection (REFLECTION) is intuitively appealing because it is valid in most models, in particular in the set-theoretic interpretation of type theory, and also because it leads to a single notion of equality in type theory. A serious disadvantage of extensional type theory is that judgemental equality and as a consequence type checking, i.e. the question whether a given judgement $\Gamma \vdash M = N : \sigma$ or $\Gamma \vdash M : \sigma$ holds, are undecidable. Intuitively, this is so, because a syntax-directed decision procedure for these judgements would have to "guess" the proof term P in the premise to rule REFLECTION. As any Π^0_1 statement in Heyting arithmetic can be encoded as a certain identity type, inhabitation of identity types is at least as complex as provability of such statements. A rigorous proof of undecidability using recursively inseparable sets is given in [6]. Therefore, extensional type theory is not really in line with the Curry-Howard isomorphism because terms do not correspond to proofs, only typing derivations do. Other shortcomings of extensional type theory are that in the presence of a universe U non-normalising terms become typeable in inconsistent contexts (for example the fixpoint combinator may be typed in $d{:}\,U, p{:}\,Id_U(d, d \to d)$) and—of a more aesthetic nature—that rule REFLECTION does not fit into the pattern of introduction and elimination rules. See, however, [10] for a formulation of REFLECTION as an η-like rule.

Probably for these reasons Martin-Löf has later on restricted judgemental equality to definitional expansion (definitional equality) and replaced rules ID-UNI and REFLECTION by his famous identity elimination rule which is motivated by the view of the identity type as an inductive family with sole constructor *Refl*.

The use of this elimination principle is described in detail in [9]. It is shown there, how it entails that propositional equality satisfies a Leibniz principle, that is if $x{:}\sigma \vdash \tau[x]$ *type* and $P : Id_\sigma(M_1, M_2)$ and $N : \tau[M_1]$ then one can construct a term $Subst_{\sigma,\tau}(M_1, M_2, P, N) : \tau[M_2]$. This Leibniz principle in turn allows one to show that propositional equality is an equivalence relation, i.e. if $P : Id(M_1, M_2)$ then we can construct $Sym\,(P) : Id(M_2, M_1)$ and for $Q : Id(M_2, M_3)$ we get $Trans\,(P, Q) : Id(M_1, M_3)$. Furthermore, it entails that Id is respected by all functions, i.e. if $U : \sigma \to \tau$ then we find $Resp\,(U, P) : Id(F\ M_1, F\ M_2)$.

Surprisingly, the intensional and extensional formulations of type theory have never been compared in the literature (to the best of our knowledge). This is the purpose of the present work. We first identify two principles which are provable in the extensional theory, but not in the intensional one and then embark on the proof that extensional type theory is conservative over the intensional one with these two principles added. We only sketch the formal argument here and refer the reader to [6] for a detailed proof.

We use a monomorphic presentation of type theory, i.e. terms with explicit type annotations, as in [9, Part III], however, without using a so-called Logical Framework, see also Sect. 3.

The material in this paper is mostly taken from the author's PhD dissertation [6]. However, some simplifications (in particular the assumption of η-conversion for Π-types) have been made, and the example in Section 2 is new.

1.1 Functional extensionality

Suppose that $\Gamma \vdash U, V : \Pi x{:}\sigma.\tau$ and $\Gamma, x{:}\sigma \vdash P : Id_\tau(U\ x\ , V\ x)$. In extensional type theory we can conclude $\Gamma \vdash Id_{\Pi x:\sigma.\tau}(U, V)$ *true* using REFLECTION, the congruence rule for λ, and η. In intensional type theory this is, however, in general not possible. A formal semantic argument for this may be found in [10]; intuitively one may argue that otherwise in the special case where Γ is empty we could deduce $Id_{\Pi x:\sigma.\tau}(U, V)$ *true* (from the existence of P above), but an identity type in the empty context can only be inhabited by a canonical element $Refl(-)$, so U and V must be definitionally equal, which does not follow from the existence of the proof P which may have been obtained using induction. To achieve this principle called *functional extensionality* we add to intensional type theory a family of constants $Ext_{\sigma,\tau}(U, V, P)$ obeying the rule

$$\frac{\begin{array}{c}\Gamma \vdash U, V : \Pi x{:}\sigma.\tau \\ \Gamma, x{:}\sigma \vdash P : Id_\tau(U\ x, V\ x)\end{array}}{\Gamma \vdash Ext_{\sigma,\tau}(U, V, P) : Id_{\Pi x:\sigma.\tau}(U, V)}\ \text{EXT-FORM}$$

1.2 Uniqueness of identity

Let $M, N : \sigma$ and $P, Q : Id_\sigma(M, N)$ be two proofs that M, N are propositionally equal. In extensional type theory it is an immediate consequence of ID-UNI that $Id_{Id_\sigma(M,N)}(P, Q)$ *true*. Streicher has conjectured in [10] that this is in general not the case in intensional type theory and in [7] this has actually been proved by a semantic argument. So again, we are led to extend our axiomatisation of propositional equality in an intensional setting. It turns out that it is enough to consider the case where Q is in canonical form so that we introduce a family of constants

$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash P : Id_\sigma(M, M)}{\Gamma \vdash IdUni_\sigma(M, P) : Id_{Id_\sigma(M,M)}(P, Refl_\sigma(M))} \quad \text{ID-UNI-I}$$

to achieve *uniqueness of identity*. Axioms like EXT-FORM and ID-UNI-I introduce non-canonical elements of all types in the empty context (not only of identity types). The constant ID-UNI-I can be endowed with an obvious reduction rule which eliminates all of its instances in closed terms of basic type. Things are more serious with EXT-FORM since no reasonable reduction rule for these constants is known. However, in [5] we discuss a possibility for eliminating these non-canonical elements by translating the identity type into suitable equivalence relations. For the present work we shall simply regard *Ext* and *IdUni* as families of constants.

1.3 The intensional type theory TT_I

It is also noticed in [10] that in the presence of uniqueness of identity the elimination operator for *Id* can be defined in terms of its particular instance *Subst* mentioned above. Therefore, we shall take *Subst* as a primitive and define *intensional type theory* (TT_I for short) as dependent type theory with rules ID-FORM, ID-INTRO, EXT-FORM, ID-UNI-I, and the following two rules for *Subst*.

$$\frac{\begin{array}{l}\Gamma \vdash P : Id_\sigma(M_1, M_2) \\ \Gamma, x{:}\sigma \vdash \tau[x]\ type \qquad \Gamma \vdash N : \tau[M_1]\end{array}}{\Gamma \vdash Subst_{\sigma,\tau}(M_1, M_2, P, N) : \tau[M_2]} \quad \text{LEIBNIZ}$$

$$\frac{\Gamma \vdash Subst_{\sigma,\tau}(M, M, Refl_\sigma(M), N) : \tau[M]}{\Gamma \vdash Subst_{\sigma,\tau}(M, M, Refl_\sigma(M), N) = N : \tau[M]} \quad \text{LEIBNIZ-COMP}$$

Recall that TT_E refers to extensional type theory defined by rules ID-UNI and REFLECTION above. Recall also that from *Subst* one may define operators *Sym*, *Trans*, and *Resp*, witnessing that *Id* is an equivalence relation compatible with function application.

In order to distinguish the intensional from the extensional type theory we write $\vdash_I$ and $\vdash_E$ for the two respective judgement relations.

The operators of TT_I are definable in TT_E, for example *Subst* is simply the identity in TT_E. More formally, we can define a "stripping map" $|-|$ by

$|\mathit{Subst}_{\sigma,\tau}(M_1, M_2, P, N)| := |N|$
$|\mathit{IdUni}_{\sigma}(M, P)| := \mathit{Refl}_{\mathit{Id}_{|\sigma|}(|M|,|M|)}(\mathit{Refl}_{|\sigma|}(|M|))$
$|\mathit{Ext}_{\sigma,\tau}(U, V, P)| := \mathit{Refl}_{\Pi x:|\sigma|.|\tau|}(|U|)$
homomorphically extended to all other terms, types, contexts, and judgements

Notice that $\Gamma = |\Gamma|$ if Γ does not contain *Subst*, *Ext*, or *IdUni*. This mapping enjoys the following trivial soundness property

Proposition 1 *If $\Gamma \vdash_I J$ then $|\Gamma| \vdash_E |J|$ for all contexts Γ and judgements J.*

2 Conservativity of TT_E over TT_I

We are now interested in a converse of the above proposition; more precisely, we shall establish the following conservativity property.

Theorem 2 *If $\Gamma \vdash_I \sigma$ type and $|\Gamma| \vdash_E M : |\sigma|$ for some M then there exists M' such that $\Gamma \vdash_I M' : \sigma$.*

Before sketching the proof of this theorem we shall illustrate its strength by an example suggested by Thomas Schreiber. The crux of this example is to show how the lack of equality reflection in TT_I can be inconvenient when one wants to actually program with dependent types rather than use type dependency merely to express constructive predicate logic.

We assume a type Bool with two canonical elements true : Bool and false : Bool. We use the following slightly unconventional elimination rule which is expressible in terms of the usual one, see [9, Ch. 21]

$$\frac{\begin{array}{l}\Gamma, b{:}\mathsf{Bool} \vdash \sigma \ \mathit{type} \\ \Gamma \vdash B : \mathsf{Bool} \\ \Gamma, p{:}\, \mathit{Id}_{\mathsf{Bool}}(B, \mathsf{true}) \vdash T : \sigma[\mathsf{true}/b] \\ \Gamma, p{:}\, \mathit{Id}_{\mathsf{Bool}}(B, \mathsf{false}) \vdash E : \sigma[\mathsf{false}/b]\end{array}}{\Gamma \vdash \mathsf{if}'_{[b:\mathsf{Bool}]\sigma}(B, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(B, \mathsf{true})]T, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(B, \mathsf{false})]E) : \sigma[B/b]} \quad \textsc{Bool-E}$$

Notice that the free variables b in σ and p in T, E, respectively, become bound in the if′ expression. This elimination operator comes with the following two computation rules:

$$\frac{\Gamma \vdash \mathsf{if}'_{[b:\mathsf{Bool}]\sigma}(\mathsf{true}, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{true}, \mathsf{true})]T, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{true}, \mathsf{false})]E) : \sigma[\mathsf{true}/b]}{\begin{array}{l}\Gamma \vdash \mathsf{if}'_{[b:\mathsf{Bool}]\sigma}(\mathsf{true}, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{true}, \mathsf{true})]T, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{true}, \mathsf{false})]E) = \\ \quad T[\mathit{Refl}_{\mathsf{Bool}}(\mathsf{true})/p] : \sigma[\mathsf{true}/b]\end{array}} \quad \textsc{Bool-C-T}$$

$$\frac{\Gamma \vdash \mathsf{if}'_{[b:\mathsf{Bool}]\sigma}(\mathsf{false}, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{false}, \mathsf{true})]T, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{false}, \mathsf{false})]E) : \sigma[\mathsf{false}/b]}{\begin{array}{l}\Gamma \vdash \mathsf{if}'_{[b:\mathsf{Bool}]\sigma}(\mathsf{false}, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{false}, \mathsf{true})]T, [p{:}\, \mathit{Id}_{\mathsf{Bool}}(\mathsf{false}, \mathsf{false})]E) = \\ \quad E[\mathit{Refl}_{\mathsf{Bool}}(\mathsf{false})/p] : \sigma[\mathsf{false}/b]\end{array}} \quad \textsc{Bool-C-F}$$

Suppose that we are given a type Loc of locations together with an equality function $\mathsf{eq} : \mathsf{Loc} \to \mathsf{Loc} \to \mathsf{Bool}$ and a proof term

$$\mathsf{eq\_correct} : \Pi l, l' : \mathsf{Loc}.\, Id_{\mathsf{Loc}}(l, l') \leftrightarrow Id_{\mathsf{Bool}}(\mathsf{eq}\ l\ l', \mathsf{true})$$

witnessing that eq is indeed an equality function on Loc. The terms eq and eq_correct may either be implemented or declared in some ambient context Γ. Furthermore, we assume a type Data depending on Loc, i.e. we have $\Gamma \vdash \mathsf{Data}(M)$ *type* whenever $\Gamma \vdash M : \mathsf{Loc}$. We make the definition

$$\mathsf{Store} = \Pi l : \mathsf{Loc}.\mathsf{Data}(l)$$

and our aim is to find a term

$$\mathsf{update} : \Pi l_0 : \mathsf{Loc}.\Pi d : \mathsf{Data}(l_0).\Pi s : \mathsf{Store}.\mathsf{Store}$$

which given l_0, d, s returns a store whose value at l_0 equals d and $(s\ l)$ at $l \neq l_0$. In TT_E we can easily construct such an update function by

$$\begin{array}{l}\mathsf{update} = \lambda l_0 : \mathsf{Loc}.\lambda d : \mathsf{Data}(l_0).\lambda s : \mathsf{Store}.\lambda l : \mathsf{Loc}. \\ \quad \mathsf{if}'_{[b:\mathsf{Bool}]\mathsf{Data}(l)}(\mathsf{eq}(l_0, l)\ , \\ \qquad [p : Id_{\mathsf{Bool}}(\mathsf{eq}(l_0, l), \mathsf{true})]d\ , \\ \qquad [p : Id_{\mathsf{Bool}}(\mathsf{eq}(l_0, l), \mathsf{false})](s\ l))\end{array}$$

The important point here is that in TT_E we have $\mathsf{Data}(l_0) = \mathsf{Data}(l)$ in the presence of $p : Id(\mathsf{eq}(l_0, l), \mathsf{true})$ by virtue of eq_correct. Therefore, we have $d : \mathsf{Data}(l)$ and the first branch of the if′ expression type-checks.

Write

$$\mathsf{update\_type} = \Pi l_0 : \mathsf{Loc}.\Pi d : \mathsf{Data}(l_0).\Pi s : \mathsf{Store}.\mathsf{Store}$$

and for $f : \mathsf{update\_type}$

$$\begin{array}{l}\mathsf{update\_spec}(f) = \\ \quad (\Pi l_0 : \mathsf{Loc}.\Pi d : \mathsf{Data}(l_0).\Pi s : \mathsf{Store}.Id_{\mathsf{Data}(l_0)}(f\ l_0\ d\ s\ l_0\ , d)) \times \\ \quad (\Pi l_0 : \mathsf{Loc}.\Pi d : \mathsf{Data}(l_0).\Pi l : \mathsf{Loc}.(Id_{\mathsf{Loc}}(l, l_0) \to \bot) \to Id_{\mathsf{Data}(l)}(f\ l_0\ d\ s\ l\ , s\ l))\end{array}$$

where $\bot$ is the empty type. We can readily establish

$$\Gamma \vdash_E \mathsf{update\_spec}(\mathsf{update})\ \mathit{true}$$

using if′ and equality reasoning.

Now assuming that Loc and Data already make sense in intensional type theory we can apply the conservativity theorem to the type

$$\sigma = \Sigma f : \mathsf{update\_type}.\mathsf{update\_spec}(f)$$

giving us an update function in TT_I which also satisfies update_spec. Coming up with such a function is not too difficult, indeed we may choose

$$\begin{array}{l}\mathsf{update'} = \lambda l_0{:}\mathsf{Loc}.\lambda d{:}\mathsf{Data}(l_0).\lambda s{:}\mathsf{Store}.\lambda l{:}\mathsf{Loc}.\\ \quad \mathsf{if}'_{[b:\mathsf{Bool}]\mathsf{Data}(l)}(\mathsf{eq}(l_0,l)\ ,\\ \qquad [p{:}\,Id_{\mathsf{Bool}}(\mathsf{eq}(l_0,l),\mathsf{true})]Subst_{\,\mathsf{Loc},\mathsf{Data}}(l_0,l\ ,((\mathsf{eq\_correct}\ l_0\ l).2\ p)\ ,\ d)\ ,\\ \qquad [p{:}\,Id_{\mathsf{Bool}}(\mathsf{eq}(l_0,l),\mathsf{false})](s\ l))\end{array}$$

where the projection .2 yields the $\leftarrow$-part of eq_correct, hence the first argument to *Subst* is of type $Id_{\mathsf{Loc}}(l_0,l)$. The conservativity theorem does not necessarily give us this very update function, but we can also use it to obtain a proof of correctness of the latter. To that end we consider the type

$$\sigma = \mathsf{update\_spec}(\mathsf{update'})$$

We have $|\sigma| = \mathsf{update\_spec}(\mathsf{update})$ so $\Gamma \vdash_E |\sigma|\ \mathit{true}$ and hence $\Gamma \vdash_I \sigma\ \mathit{true}$ giving us the desired correctness proof for our particular function update'. Constructing such proof directly requires quite some effort as the function in question contains an instance of *Subst*. In this case the author has been able to construct such a proof using the Lego system. This revealed that the use of *Ext* can be avoided here and that *IdUni* is only used for the particular type Loc. As observed by Michael Hedberg *IdUni* is definable from Martin-Löf's elimination rule for types with decidable equality (such as Loc) so that this example can be carried out in pure intensional Martin-Löf type theory without the additional constants *Ext* and *IdUni*.

It was pointed out by one of the referees that in the presence of pattern-matching like in ALF [1] the complicated equality reasoning in the example can be avoided if one replaces eq and eq_correct by a single constant

$$\mathsf{eq}' : \Pi l, l'{:}\mathsf{Loc}.Id_{\mathsf{Loc}}(l,l') + (Id_{\mathsf{Loc}}(l,l') \to \bot)$$

One may then define update by pattern matching over $\mathsf{eq}'(l,l_0)$ and in the positive case we may then assume that l and l_0 are *judgementally* equal. It is not clear, however, how this technique can be applied in general, in particular, the example in [6] and briefly mentioned in Section 3 does not seem to be amenable to a simple development using pattern-matching.

Proof of Thm. 2 (Sketch). The proof is based on two main ingredients. First, we observe that a consequence of this conservativity property is that types σ and σ' which are equal in TT_E must be isomorphic in TT_I. To see this consider the type

$$\begin{array}{l}\rho := \Sigma f{:}\sigma \to \sigma'.\Sigma f^{-1}{:}\sigma' \to \sigma.\\ \qquad Id(\lambda x{:}\sigma.f^{-1}(f\ x),\lambda x{:}\sigma.x) \times Id(\lambda x{:}\sigma'.f(f^{-1}\ x),\lambda x{:}\sigma'.x)\end{array}$$

If $\vdash_E |\sigma| = |\sigma'|$ then $|\rho|$ is inhabited by two instances of the identity function together with two instances of reflexivity. So a proof of Theorem 2 must embody a construction of these isomorphisms. Now the only way to deduce judgemental type equalities in TT_E is by using congruence rules and REFLECTION in the case of type formers containing terms like the identity type. So we can compute these

isomorphisms directly by induction on the structure of any two types σ and σ' without using conservativity in the first place.

The second ingredient is a general method for establishing conservativity between type theories. Generalising from Thm. 2 we call an extension $\mathbf{T}'$ of some type theory $\mathbf{T}$ *conservative* over $\mathbf{T}$ if whenever a type of $\mathbf{T}$ is inhabited in $\mathbf{T}'$ then it is already inhabited in $\mathbf{T}$.

Proposition 3 *Let $\mathbf{T}$ be a type theory and $\mathbf{T}'$ an extension of $\mathbf{T}$. $\mathbf{T}'$ is conservative over $\mathbf{T}$ if and only if there exists a model $\mathbf{Q}$ of $\mathbf{T}'$, hence of $\mathbf{T}$, such that the interpretation of $\mathbf{T}$ in this model is full (surjective).*

Proof. Suppose that σ is a type of $\mathbf{T}$ such that σ is inhabited in $\mathbf{T}'$ by some term M. The interpretation of M in $\mathbf{Q}$ yields a "semantic term" of the interpretation of σ in $\mathbf{Q}$ which by fullness gives a term of type σ in $\mathbf{T}$. For the converse we let $\mathbf{Q}$ be the term model of $\mathbf{T}'$

At this level of generality the argument is too informal so as to be of direct use. It only serves us as a guideline which the reader is invited to bear in mind. The theories $\mathbf{T}$ and $\mathbf{T}'$ are TT_I and TT_E, respectively, where a slight complication results from the fact that TT_E is not literally an extension of TT_I, but that we have an interpretation of one in the other, namely the stripping map. The role of "models" is played by *categories with families* in the sense of Dybjer [2, 4] or any other category-theoretic notion of model for type theory. For the purpose of this abstract it suffices to know that such a structure provides domains of interpretation for contexts, types, and terms[1], and comes with semantic type and term formers operating on these semantic objects. An interpretation function can then be defined which maps syntactic contexts, types, terms to their semantic companions and translates syntactic type and term formers into the corresponding semantic ones. Judgemental equality is then modelled by set-theoretic equality of semantic objects. Factoring the syntax by judgemental equality yields a particular model: the term model. The interpretation of the syntax in a some model induces a unique structure preserving map from the term model to this model. The model $\mathbf{Q}$ we use is a quotient of the term model of TT_I by propositional equality where in addition we identify types and contexts which are canonically isomorphic in the sense described above, i.e. which become equal in TT_E.

Let us now look at some of the details. In order to define the canonical isomorphisms between types one also needs to consider isomorphisms between contexts to get the inductive definition through. In order to specify those we extend propositional equality to contexts and context morphisms (substitutions) using Σ-types. It turns out that for this extension the same combinators as for ordinary propositional equality can be defined. Then by simultaneous induction on contexts and types we construct possibly undefined "isomorphism candidates" between any two contexts and types. That is, for contexts Γ, Δ we have a (possibly undefined) context morphism (a $|\Delta|$-tuple of terms) $co_{\Gamma,\Delta} : \Gamma \to \Delta$

[1] ...and substitutions, but we gloss over this point here.

which if defined possesses an inverse up to propositional equality. For types $\Gamma \vdash_I \sigma\ type$, $\Delta \vdash_I \tau\ type$ we have a (possibly undefined) term

$$\Gamma, x{:}\sigma \vdash_I ty_{\Gamma,\Delta,\sigma,\tau}[x] : \tau[co_{\Gamma,\Delta}]$$

which—if defined—is an isomorphism w.r.t. propositional equality. We use square brackets to denote (generalised) substitution, i.e. if $f : \Gamma \to \Delta$ and $\Delta \vdash \sigma\ type$ then $\Gamma \vdash \sigma[f]\ type$ and similarly for other judgements. The isomorphism between contexts of different length is always undefined and between contexts of the same length it is obtained from the isomorphisms between types. An isomorphism between types can only be defined if they share the same outermost type former. In the case of Π and Σ-types the isomorphism is the obvious lifting of the isomorphism for the respective components. Functional extensionality is used to verify the isomorphism property in the case of Π-types. The interesting case is where σ and τ are identity types. For simplicity we assume that they are both in the same context and over the same type. I.e. let $\Gamma \vdash_I M_1, M_2, N_1, N_2 : \rho$ and put $\sigma = Id_\rho(M_1, M_2)$ and $\tau = Id_\rho(N_1, N_2)$. The isomorphism

$$\Gamma, p{:}\, Id_\rho(M_1, M_2) \vdash_I ty_{\Gamma,\Gamma,\sigma,\tau}[p] : Id_\rho(N_1, N_2)$$

is defined iff M_1, N_1 and M_2, N_2 are propositionally equal. In this case we pick $\Gamma \vdash_I P_1 : Id_\rho(M_1, N_1)$ and $\Gamma \vdash_I P_2 : Id_\rho(M_2, N_2)$ and set

$$ty_{\Gamma,\Gamma,\sigma,\tau}[p] := Trans\,(Trans\,(Sym\,(P_1), p), P_2)$$

Notice that in this case ty is defined w.r.t. some arbitrary choice of these proofs P_i. The isomorphism property is readily established using *IdUni*.

Having defined these isomorphisms we construct a model of TT_E in which

- contexts are equivalence classes of contexts, two contexts being identified if the isomorphism between them is defined,
- types in context $[\Delta]_\sim$ are equivalence classes of pairs (Γ, σ) where $\Gamma \in [\Delta]_\sim$, and $\Gamma \vdash_I \sigma\ type$, and two such pairs are identified if the corresponding ty-isomorphism is defined,
- terms of type $[(\Delta, \tau)]_\sim$ are equivalence classes of triples (Γ, M, σ) where $(\Gamma, \sigma) \in [(\Delta, \tau)]_\sim$, and $\Gamma \vdash_I M : \sigma$, and two such triples (Γ, M, σ) and (Γ', N', σ') are identified if

$$\Gamma \vdash Id_{\sigma'}(ty_{\Gamma,\Gamma',\sigma,\sigma'}[M]\ ,\ M'[co_{\Gamma,\Gamma'}])\ true$$

  i.e., if M and M' with suitably adjusted source and target are propositionally equal in TT_I.

We have used here the notation $[-]_\sim$ for equivalence classes associated to a representative.

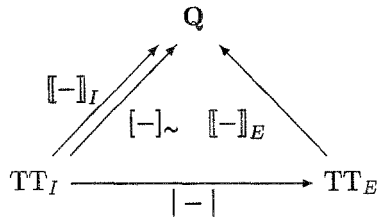
Now we prove that this structure, which we call $\mathbf{Q}$, does indeed form a model of TT_E and that in particular the required semantic type and term formers are given by applying their syntactic companions (in TT_I) to equivalence classes. Here *Ext* and *IdUni* are used to show that various settings are independent of

the choice of representatives and thus are well-defined on equivalence classes. For example, if $[(\Gamma, M, \sigma)]_\sim$ and $[(\Gamma', N', \sigma')]_\sim$ are two semantic terms of (semantic) type $[(\Delta, \sigma)]_\sim$ then we define the associated identity type as the equivalence class of the pair of Γ and

$$\Gamma \vdash Id_{\sigma'}(ty_{\Gamma,\Gamma',\sigma,\sigma'}[M]\ ,\ M'[co_{\Gamma,\Gamma'}])\ type$$

Since we have defined equality of terms in $\mathbf{Q}$ precisely as inhabitedness of this type the rule REFLECTION is valid in $\mathbf{Q}$. Uniqueness of identity is required here to show that this setting is independent of the chosen representatives (Γ, M, σ) and (Γ', N', σ').

The above can be summarised by the following diagram of structure preserving maps between models.



where TT_I and TT_E refer to the term models of the respective type theories. $[\![-]\!]_I$ and $[\![-]\!]_E$ denote the interpretations of TT_I and TT_E in $\mathbf{Q}$, which being a model for TT_E also models TT_I. Finally, $[-]_\sim : \mathrm{TT}_I \to \mathbf{Q}$ is the projection map associating equivalence classes which is structure preserving by definition of $\mathbf{Q}$.

By induction on derivations or—more elegantly—by an initiality argument we find that all three structure preserving maps from TT_I to $\mathbf{Q}$ are equal, more precisely, we have

$$[-]_\sim = [\![-]\!]_I = [\![-]\!]_E \circ |-| \tag{1}$$

Intuitively, we can now argue that the interpretation of TT_I in $\mathbf{Q}$ is surjective because we have characterised it as the projection associating equivalence classes and then apply Prop. 3. More formally, suppose that $\Gamma \vdash_I \sigma$ and $|\Gamma| \vdash_E M : |\sigma|$. Let $[\![M]\!]_E = [(\Gamma', M', \sigma')]_\sim$ be the interpretation of M in $\mathbf{Q}$. By definition, we have $\Gamma' \vdash_I M' : \sigma'$. Moreover, by Eqn. 1 we find

$$[\Gamma]_\sim = [\![|\Gamma|]\!]_E = [\Gamma']_\sim$$

and

$$[(\Gamma, \sigma)]_\sim = [\![|(\Gamma, \sigma)|]\!]_E = [(\Gamma', \sigma')]_\sim$$

This in turn implies that $co_{\Gamma,\Gamma'}$ and $ty_{\Gamma,\Gamma',\sigma,\sigma'}$ are defined. Therefore, we have

$$\Gamma \vdash_I ty^{-1}{}_{\Gamma,\Gamma',\sigma,\sigma'}[M'[co_{\Gamma,\Gamma'}]] : \sigma$$

giving the desired inhabitant of σ in TT_I. □

We remark that we can even show that the stripping of the thus constructed inhabitant of σ is equal to M in TT_E by showing that the stripping map respects the equality in $\mathbf{Q}$ and thus lifts to a structure preserving map from $\mathbf{Q}$ to TT_E which again by initiality must be a left inverse to $[\![-]\!]_E$. We refer to [6] for details.

3 Discussion

Although relatively complex, the described method is fairly robust w.r.t. extensions of the type theory. For it to be extensible to a new type former it is enough that this type former admits an action on propositional isomorphisms. In [6] we demonstrate this by adding quotient types and a universe. We remark, however, that this condition is not met in the presentation of type theory using the Logical Framework in the sense of [9, Part III]. The reason is that there we can have variables of type $Set \to Set$ which act as completely unspecified type formers. Nevertheless, it is plausible that Thm. 2 continues to hold in this case. A possible proof could use a one-step definition of the isomorphism candidates by which for example the isomorphism $ty_{\Gamma,\Gamma,\sigma,\tau}$ would be defined iff there exists a context Δ, a type ρ, and context morphisms $f, g : \Gamma \to \Delta$ such that $\Gamma \vdash_I \sigma = \rho[f]$, $\Gamma \vdash_I \tau = \rho[g]$, and f, g are (component-wise) propositionally equal. We leave the details to a future paper.

We also point out the non-constructive nature of the proof. Not only has the axiom of choice been used in the definition of the canonical isomorphisms *co* and *ty*; more seriously the interpretation of TT_E in $\mathbf{Q}$ associates equivalence classes to contexts, types, and terms. In order to get an inhabitant of type σ in the proof above we must arbitrarily choose a representative of the corresponding class. We have done so implicitly by writing $[\![M]\!]_E = [(\Gamma', M', \sigma')]_\sim$. So the present proof does not directly give rise to an algorithm which effectively computes an inhabitant of σ from a derivation of $|\sigma|$ *true* in TT_E. Now such algorithm trivially exists by Markov's principle, that is we simply try out all possible terms and derivations and from the non-constructive proof of existence we know that this search always succeeds. But of course, one would like a more efficient algorithm which makes use of the given derivation in TT_E. It is, however, not clear whether the described argument gives rise to one such. An idea would be to carry out the construction of $\mathbf{Q}$ in some "setoid model" where quotients come with a canonical choice of representatives. This is, however, not possible using a semantics which interprets judgemental equality as set-theoretic extensional equality as we have done.

It appears that a constructivisation of the result could give rise to tactics facilitating theorem proving in TT_I, the idea being that parts of a proof or program development could be carried out in TT_E. The resulting derivation would then be automatically translated into TT_I. In [6] we give a proof in TT_E of a theorem of Mendler's stating that the dependent sum (Σ) preserves limits of ω-chains. Although the proof in TT_E is relatively straightforward it was despite considerable effort so far not possible to obtain a fully formal development in TT_I. The reason was that in this case one has to prove a property about a term which contains instances of *Subst* inside a primitive recursion over natural numbers rather than just booleans as in the case of the update function above. Given sufficient time and energy one could certainly push this proof through, but a systematic treatment based on the conservativity result would seem to save much work in such cases.

We remind the reader that the conservativity result applies to intensional

type theory with functional extensionality and uniqueness of identity added. Another issue is the conservativity of the latter two over pure intensional type theory or rather the characterisation of those types and contexts for which they are conservative. This question remains unanswered and is left for future research. A brief discussion of the topic may be found in [6].

Judgemental equality has played a minor role in the present proof and it appears that the whole development would go through if TT_I was replaced by a type theory without judgemental equality at all and rules like β or LEIBNIZ-COMP replaced by corresponding constants of identity types in the style of IDUNI. The construction of the model $\mathbf{Q}$ would remain unchanged since propositionally equal objects are identified in $\mathbf{Q}$, some care has to be taken, however, with the extension of propositional equality to contexts.

This generalisation would also answer affirmatively the question of conservativity of the original TT_I over this new type theory without judgemental equality.

Acknowledgement. I wish to thank Thomas Streicher and the anonymous referees for suggesting improvements of an earlier draft.

References

1. Thorsten Altenkirch, Veronica Gaspes, Bengt Nordström, and Björn von Sydow. *A User's Guide to ALF.* Chalmers University of Technology, Sweden, May 1994. Available on the WWW: file://ftp.cs.chalmers.se/pub/users/alti/alf.ps.Z.
2. Peter Dybjer. Internal type theory. In *Proc. BRA TYPES workshop, Torino, June 1995, Springer LNCS*, 1996. To appear.
3. Solomon Feferman. Theories of finite type. In Jon Barwise, editor, *Handbook of Mathematical Logic*, chapter D.4. North-Holland, 1977.
4. Martin Hofmann. Syntax and sematnics of dependent types. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation.* Cambridge University Press, 199?
5. Martin Hofmann. Elimination of extensionality for Martin-Löf type theory. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programs.* Springer, 1994. LNCS 806.
6. Martin Hofmann. *Extensional Concepts in Intensional Type Theory.* PhD thesis, Univ. of Edinburgh, 1995.
7. Martin Hofmann and Thomas Streicher. A groupoid model refutes uniqueness of identity proofs. In *Proceedings of the 9th Symposium on Logic in Computer Science (LICS), Paris*, 1994.
8. Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis·Napoli, 1984.
9. B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory, An Introduction.* Clarendon Press, Oxford, 1990.
10. Thomas Streicher. *Semantical Investigations into Intensional Type Theory.* Habilitationsschrift, LMU München, 1993.