A Simple Model for Quotient Types

Martin Hofmann*

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

Abstract. We give an interpretation of quotient types within in a dependent type theory with an impredicative universe of propositions (Calculus of Constructions). In the model, type dependency arises only at the propositional level, therefore universes and large eliminations cannot be interpreted. In exchange, the model is much simpler and more intuitive than the one proposed by the author in [10]. Moreover, we interpret a choice operator for quotient types that, under certain restrictions, allows one to recover a representative from an equivalence class. Since the model is constructed syntactically, the interpretation function from the syntax with quotient types to the model gives rise to a procedure which eliminates quotient types by replacing propositional equality by equality relations defined by induction on the type structure ("book equalities").

1 Introduction

Intensional type theories like the Calculus of Constructions have been proposed as a framework in which to formalise mathematics and in which to specify and verify computer programs. Mathematical reasoning and to a lesser extent program specification make use of the possibility to redefine equality on a set by quotienting. Examples are the definition of integers as a quotient of pairs of natural numbers or of finite sets as a quotient of lists. In intensional type theories there is no such facility, which is why when formalising constructions involving quotients one is forced to replace equality on each type by an equivalence relation and to prove that all the functions and predicates one defines preserve these relations. This leads to complicated proofs and large bookkeeping requirements.

We claim that this flaw can be remedied. The key idea is to introduce operators for quotienting and other extensional concepts and to give a translation of the type theory including these operators into the pure type theory, under which types get translated into types together with a (partial) equivalence relation. One may then either print out the translation (for example if for conceptual reasons one is interested in a derivation using partial equivalence relations instead of quotient types.) or hide it and only use it in order to decide definitional equality in the theory with the quotient operators.

In [10] we have given such an interpretation[2] for Martin-Löf type theory without impredicative propositions. The interpretation given there is fairly complicated because it accounts for type dependency in full generality. If we confine

* Supported by a EU HCM fellowship, contract number ERBCHBICT930420

[2] The interpretation of quotient types is not explicitly carried out in [10].

the basic source of type dependency to the dependency of a type of proofs on the corresponding proposition, i.e. $x{:}\mathrm{Prop} \vdash \mathrm{Prf}(x)$ **type** then a much simpler interpretation may be given. For many applications, such a more restrained discipline of type dependency is sufficient. See also Section 6.

We translate a type theory with quotient types and a proof-irrelevant universe of propositions into the pure Calculus of Constructions with inductive types. Types get interpreted as types together with partial equivalence relations ("setoids"), terms get interpreted as terms together with proofs that the relations are respected. As opposed to the approach in [10] type dependency is only interpreted at the level of the relations — the underlying types in the interpretation are all fixed, i.e. do not depend on their context. This not only gives the desired interpretation of quotient types, but also leads to a full separation between proofs and programs which may be interesting to study in its own right.

Related work. Quotient types are available in extensional frameworks without explicit proof objects like Nuprl [7] and HOL [9] and also in the internal language of toposes [16]. The rules given there are quite similar to the ones proposed here. For an intensional calculus with explicit proof objects no extension with quotient types exists in the literature to our knowledge. Categorical formulations of extensional quotient types in simple type theory have recently been studied by Jacobs [11] without giving particular instances of models, though.

The method of interpreting type theory in itself so as to obtain additional features has also been used by Martin-Löf in order to account for proof irrelevance and subset types [15]. The use of categorical model theory (Section 3) to describe such translations appears to be new. Also the choice operator we introduce seems original and hinges on the intensional nature of definitional equality.

The model we give bears some resemblance with realizability interpretations of type theory [3] in which types are interpreted as partial equivalence relations on the natural numbers. Again, the main difference is that these models are extensional and therefore semantic equality is undecidable and the choice operator cannot be interpreted. Also these models do not give rise to a translation of type theory into itself, but only to an assignment of untyped algorithms to proofs.

2 Syntax

We consider a source type theory and a target type theory. The target type theory is a version of the Calculus of Constructions with a type of natural numbers and optionally more inductive types. Its (entirely standard) syntax is given in Appendix A. The reader who is not familiar with the Calculus of Constructions and related systems may want to take a look at this Appendix before reading on. Let us here just summarise the form of the judgements of the target type theory and some peculiarities. A context is a well-formed list of variable declarations $\Gamma = x_1{:}\sigma_1, \ldots, x_n{:}\sigma_n$ where due to type dependency the variables x_i may appear free in σ_j if $i < j \leq n$. The judgements are Γ **ctxt** to mean that

Γ is a well-formed context of variable declarations, $\Gamma \vdash \sigma$ **type** to mean that σ is a type in context Γ, $\Gamma \vdash M : \sigma$ to mean that M is a term of type σ in context Γ, $\Gamma = \Delta$ **ctxt** to mean that Γ and Δ are definitionally equal contexts, $\Gamma \vdash \sigma = \tau$ **type** to mean that σ and τ are definitionally equal types in context Γ, and $\Gamma \vdash M = N : \sigma$ to mean that M and N are definitionally equal terms of type σ in context Γ. So we have a typed definitional equality which is given inductively by a system of rules. It is known (see e.g. [6, 1]) that there is a decision procedure for it using untyped conversion, which is often regarded as the very *definition* of definitional equality. Another thing which is slightly unconventional is that if M : Prop we have $\mathrm{Prf}(M)$ **type** rather than M **type** as in the PTS-tradition [2]. So an expression is either a term or a type but never both.

The source type theory is meant to be the internal language of the model which we are going to describe. It supports all the type and term formers and rules present in the target type theory, but in addition provides quotient types. These are defined by the following rules.

$$\frac{\Gamma \vdash \sigma \qquad \Gamma\ ,\ s,s' {:}\sigma \vdash R[s,s'] : \mathrm{Prop}}{\Gamma \vdash \sigma/R\ \mathbf{type}}\ \text{Q-Form} \qquad \frac{\Gamma \vdash M {:} \sigma}{\Gamma \vdash [M]_R {:} \sigma/R}\ \text{Q-Intro}$$

$$\frac{\begin{array}{l}\Gamma \vdash \tau\ \mathbf{type} \quad \Gamma\ ,\ s{:}\sigma \vdash M[s] : \tau \qquad \Gamma \vdash N : \sigma/R \\ \Gamma\ ,\ s,s' {:}\sigma\ ,\ p{:}\mathrm{Prf}(R[s,s']) \vdash H : \mathrm{Prf}(M[s] \stackrel{L}{=} M[s'])\end{array}}{\Gamma \vdash \mathrm{plug}_R\, N \,\mathrm{in}\, M \,\mathrm{using}\, H : \tau}\ \text{Q-Elim}$$

$$\frac{\Gamma \vdash \mathrm{plug}_R\, [N]_R \,\mathrm{in}\, M \,\mathrm{using}\, H : \tau}{\Gamma \vdash \mathrm{plug}_R\, [N]_R \,\mathrm{in}\, M \,\mathrm{using}\, H = M[N] : \tau}\ \text{Q-Comp}$$

$$\frac{\begin{array}{l}\Gamma \vdash M, N : \sigma \\ \Gamma \vdash H : \mathrm{Prf}(R[M,N])\end{array}\ \text{Q-Ax}}{\Gamma \vdash \mathrm{Qax}_R(H) : \mathrm{Prf}([M]_R \stackrel{L}{=} [N]_R)} \qquad \frac{\begin{array}{l}\Gamma\ ,\ x{:}\sigma/R \vdash P[x] : \mathrm{Prop} \\ \Gamma\ ,\ s{:}\sigma \vdash H : \mathrm{Prf}(P[[s]_R]) \\ \Gamma \vdash M : \sigma/R\end{array}\ \text{Q-Ind}}{\Gamma \vdash \mathrm{Qind}_R(H,M) : \mathrm{Prf}(P[M])}$$

Notice that plug and Qind are *binders* as indicated by the typing rules. To reduce clutter we do not explicitly indicate the bound variables in these terms. This may be made precise using de Bruijn indices. The symbol $\stackrel{L}{=}$ refers propositional (Leibniz) equality, see 2.1 below. This syntax is a formalisation of usual mathematical practice. Rule Q-Form allows the formation of a type σ/R from a a relation R on some type σ. We do not require R to be an equivalence relation. Using Q-Intro one constructs elements ("classes") of σ/R from "representatives". The rule Q-Elim allows one to construct functions on the quotient type by definition on representatives. The axiom Q-Ax states that the classes of related elements are equal; the axiom Q-Ind states that σ/R consists of "classes" only. One may also consider a dependent version of Q-Elim where an instance of Q-Ax has to be inserted in order to make the proviso on compatibility typecheck. In the present setting such rule is derivable.

For example, we may define a type of integers as $\mathrm{Int} := \mathrm{Nat} \times \mathrm{Nat}/R_{\mathrm{Int}}$ where $R_{\mathrm{Int}}[u,v{:}\mathrm{Nat} \times \mathrm{Nat}] = (u.1 + v.2 \stackrel{L}{=} u.2 + v.1)$. The elimination rule now allows to define functions like addition on the integers and the induction principle

together with the equations permits to derive properties of these functions from their implementations. In examples we assume that various such functions such as $+$, $|-|$, and $-$ (negation) have been defined.

These rules differ from the ones e.g. in [11] because of the distinction between propositional equality (for equivalence classes) and definitional equality for computations. Up to propositional equality, however, Jacobs rules are derivable from ours and vice versa.

It is easy to see that an interpretation of types as sets, of the type Prop as the set $\{tt, ff\}$ and of quotient types σ/R as the set-theoretic quotient of σ by the equivalence relation generated by R validates all of the above rules. This is not the case for the choice operator we consider. It is governed by a rule of the form

$$\frac{\Gamma \vdash M : \sigma/R \qquad \text{"certain proviso"}}{\Gamma \vdash \mathrm{choice}(M) : \sigma} \quad \text{Q-Choice}$$

together with two associated equality rules

$$\frac{\Gamma \vdash M : \sigma/R \qquad \Gamma \vdash \mathrm{choice}(M) : \sigma}{\Gamma \vdash [\mathrm{choice}(M)]_R = M : \sigma/R} \quad \text{Q-Choice-Ax}$$

$$\frac{\Gamma \vdash \mathrm{choice}([M]_R) : \sigma \quad \Gamma \vdash M : \sigma}{\Gamma \vdash \mathrm{choice}([M]_R) = M : \sigma} \quad \text{Q-Choice-Comp}$$

The first rule may be interpreted set-theoretically using a fixed choice of representatives for each equivalence class. The second rule, which states that choice recovers *the* representative used to form an equivalence class, stands in contradiction to this interpretation, and in fact to any extensional understanding of quotient types. It does make sense, however, in an intensional setting in which quotient types are represented by their underlying types and only the propositional equality is redefined. The "certain proviso" roughly states that the free variables in M are not of quotiented type. This is described in Section 5.2 where we also discuss possible applications of the choice operator. It is crucial for the soundness of the choice operator that propositional equality ($\stackrel{L}{=}$) and definitional equality ($=$) are distinct. The former refers to provable extensional equality whereas the latter refers to intensionally equal "algorithms".

In addition to these quotient types the source type theory contains rules which identify all proofs of a proposition (proof irrelevance) and which define biimplication as the propositional equality on the type of propositions. Moreover, pointwise propositionally equal functions are propositionally equal and in view of proof irrelevance the Σ-type can be used as a subset type former. Thus, the source type theory approximates to a good deal naive set theory.

Definitional equality in the source type theory is defined semantically via the interpretation; we give, however, a few derived equality rules and in particular all the rules in the target type theory continue to hold. This is in line with the intuitive understanding of definitional equality as induced by definitional expansion. We view quotient types and other extensional concepts as being *defined* in terms of interpretations like the one we give. Since the model is built on syntax and (decidable) definitional equality the semantically defined equality is decidable, too.

2.1 Some Abbreviations

For types σ, τ in some context Γ we write $\sigma \to \tau$ for $\Pi x{:}\sigma.\tau$ and $\sigma \times \tau$ for $\Sigma x{:}\sigma.\tau$. If C is a term, type, or context, we write $C[x := M]$ for the syntactic capture-free substitution of M for x in C. We use logical connectives $\Rightarrow, \wedge, \Leftrightarrow, tt, ff$ for their encodings in the Calculus of Constructions. If $P, Q : \mathrm{Prop}$ then their implication $P \Rightarrow Q$ is defined as the proposition $\forall x{:}\mathrm{Prf}(P).Q$ and their conjunction $P \wedge Q$ is defined as the proposition $\forall c{:}\mathrm{Prop}.(P \Rightarrow Q \Rightarrow c) \Rightarrow c$. Bi-implication $P \Leftrightarrow Q$ is $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$. The true proposition tt is defined as $\forall c{:}\mathrm{Prop}.c \Rightarrow c$. The false proposition ff is defined as $\forall c{:}\mathrm{Prop}.\mathrm{Prf}(c)$. For $M, N{:}\sigma$ we define Leibniz equality $M \stackrel{L}{=} N$ as $\forall P{:}\sigma \to \mathrm{Prop}.(P\ M) \Rightarrow (P\ N)$ It is well-known that the proof rules for these connectives in higher-order intuitionistic logic are derivable [6]. Furthermore, we introduce the notation $\Gamma \vdash P$ true to mean that $\Gamma \vdash P{:}\mathrm{Prop}$ and that there exists a term $\Gamma \vdash M : \mathrm{Prf}(P)$. If $\Gamma \vdash P : \mathrm{Prop}$ we sometimes simply write P in the running text to mean $\Gamma \vdash P$ true.

Context morphisms. Let Γ and $\Delta = x_1{:}\tau_1, \ldots, x_n{:}\tau_n$ be contexts. A *syntactic context morphism* from Γ to Δ is an n-tuple of terms $f = (M_1, \ldots, M_n)$ with $\Gamma \vdash M_1 : \tau_1$, $\Gamma \vdash M_2 : \tau_2[x_1 := M_1]$ through to $\Gamma \vdash M_n : \tau_n[x_1 := M_1] \ldots [x_{n-1} := M_{n-1}]$. In this situation we write $f : \Gamma \Rightarrow \Delta$. Definitional equality is extended canonically to syntactic context morphisms, that is we write $f = f' : \Gamma \Rightarrow \Delta$ if f and f' have the same length and definitionally equal components. The syntactic context morphisms include the *identity* from Γ to Γ which consists of the tuple of variables in Γ and they are closed under composition, i.e. if f is a syntactic context morphism from Γ to Δ and g is a syntactic context morphism from Δ to Θ then the parallel substitution of f into g gives a context morphism from Γ to Θ. In this way the contexts form a category, i.e. composition is associative and the identity is neutral. The empty context is a terminal object in this category.

If Γ is a context of length n then if $\gamma = (\gamma_1, \ldots \gamma_n)$ is an n-tuple of variables then $\gamma{:}\Gamma$ denotes the context Γ with the variables renamed to $\gamma_1 \ldots \gamma_n$. Now assume $\Gamma, \Delta \vdash \sigma$ for some type expression σ. By the declaration $\tau[\delta{:}\Delta] := \sigma$ the meta-variable τ becomes an abbreviation for the expression σ. This notation emphasises that the variables from Δ are free in τ. For example we may define $\mathrm{eqzero}[n{:}\mathrm{Nat}] := (n \stackrel{L}{=} 0)$. Explicit variable names in a substitution may now be omitted, for example eqzero[5] denotes the proposition $5 \stackrel{L}{=} 0$. If $f : \Gamma \Rightarrow \Delta$ and $\Delta \vdash \sigma$ **type** then we write $\sigma[f]$ for the parallel substitution of f into σ. We have $\Gamma \vdash \sigma[f]$ **type** and similarly for terms.

3 Categorical Semantics

It turns out that due to type dependency the verification that a certain translation indeed validates all the rules of type theory is quite involved. It has therefore proven useful to insert the intermediate step of abstract categorical models. One then defines a sound interpretation function once and for all and the task of proving that a certain translation is sound can be reduced to the task of verifying that one has an instance of the abstract model. This may be compared to

the correspondance between typed λ-calculus and cartesian closed categories as described in [13]. The role of cartesian close categories is played here by the (by now standard) notion of *categories with attributes* introduced by Cartmell [5]. We only sketch this notion of model and the interpretation of syntax therein. The interested reader is referred to [17, 19, 12].

Definition 1 *A* category with attributes *(cwa) is given by the following data and conditions:*

- *A category* $\mathbf{C}$ *with terminal object* 1*. The objects of* $\mathbf{C}$ *are called* contexts *(*$\mathrm{A}, \mathrm{B}, \Gamma, \Delta, \ldots$*), the morphisms are called* context morphisms *(*$f, g, h, \ldots$*). The terminal object* $\mathbf{1}$ *is also called the* empty context*. The unique morphism from a context* Γ *to* $\mathbf{1}$ *is denoted* $!_\Gamma$*.*
- *A functor* $\mathrm{Fam} : \mathbf{C}^{\mathrm{op}} \to \mathbf{Sets}$*. If* $f \in \mathbf{C}(\Gamma, \Delta)$ *is a context morphism and* $\sigma \in \mathrm{Fam}(\Delta)$ *then* $\mathrm{Fam}(f)(\sigma) \in \mathrm{Fam}(\Gamma)$ *is abbreviated* $\sigma\{f\}$*. The elements of* $\mathrm{Fam}(\Gamma)$ *are called* types *or* families *(*$\sigma, \tau, \rho \ldots$*) in context* Γ*. The family* $\sigma\{f\} \in \mathrm{Fam}(\mathrm{B})$ *is called the* substitution *of* σ *along* f*.*
- *If* σ *is a family in context* Γ *then there is a context* $\Gamma \cdot \sigma$*, called the* comprehension *of* σ*, and a context morphism* $\mathrm{p}(\sigma) \in \mathbf{C}(\Gamma \cdot \sigma, \Gamma)$ *called the* canonical projection *of* σ*. Moreover, if in addition* $f \in \mathbf{C}(\mathrm{B}, \Gamma)$ *then there is a morphism* $\mathrm{q}(f, \sigma) \in \mathbf{C}(\mathrm{B} \cdot \sigma\{f\}, \Gamma \cdot \sigma)$ *and*

$$\begin{array}{ccc} \mathrm{B} \cdot \sigma\{f\} & \xrightarrow{\mathrm{q}(f,\sigma)} & \Gamma \cdot \sigma \\ \Big\downarrow{\scriptstyle \mathrm{p}(\sigma\{f\})} & & \Big\downarrow{\scriptstyle \mathrm{p}(\sigma)} \\ \mathrm{B} & \xrightarrow[f]{} & \Gamma \end{array} \qquad (\ddagger\ddagger\ddagger)$$

  is a pullback diagram.
  The assignment $\mathrm{q}(-,-)$ *is functorial in the sense that* $\mathrm{q}(\mathrm{id}_\Gamma, \sigma) = \mathrm{id}_{\Gamma\cdot\sigma}$ *and for* $g \in \mathbf{C}(\mathrm{A}, \mathrm{B})$ *also* $\mathrm{q}(f \circ g, \sigma) = \mathrm{q}(f, \sigma) \circ \mathrm{q}(g, \sigma\{f\})$*. Note that these equations "typecheck" by virtue of functoriality of* Fam*.*

An example for a cwa is the term model of some dependently typed calculus constructed as indicated by the names of the various components of a cwa. Fix a cwa for the rest of this section. Let $\sigma \in \mathrm{Fam}(\Gamma)$. A *section* of σ is a morphism $M : \Gamma \to \Gamma \cdot \sigma$ with $\mathrm{p}(\sigma) \circ M = \mathrm{id}_\Gamma$. The sections correspond to terms of σ in the syntax. If furthermore $f : \mathrm{B} \to \Gamma$ then by the universal property of pullbacks there exists a unique section $M\{f\}$ of $\sigma\{f\}$ with $\mathrm{q}(f, \sigma) \circ M\{f\} = M \circ f$. This corresponds to substitution in terms. We write $\mathrm{Sect}(\sigma)$ for the set of sections of σ. Substitution along a canonical projection corresponds to weakening. We introduce some abbreviations for this. If $\sigma, \tau \in \mathrm{Fam}(\Gamma)$, $M \in \mathrm{Sect}(\sigma)$, and $f : \mathrm{B} \to \Gamma$ we put$\sigma^+ := \sigma\{\mathrm{p}(\tau)\} \in \mathrm{Fam}(\Gamma \cdot \tau)$ and $M^+ := M\{\mathrm{p}(\tau)\} \in \mathrm{Sect}(\sigma^+)$ and $f^+ := \mathrm{q}(f, \tau) \in \mathbf{C}(\mathrm{B} \cdot \tau\{f\}\ , \Gamma \cdot \tau)$. This notation being ambiguous (e.g. f^+ can mean both $\mathrm{q}(f, \sigma)$ and $\mathrm{q}(f, \tau)$) we use it only if the meaning is clear from the context.

For $\sigma \in \mathrm{Fam}(\Gamma)$ there is a unique section v_σ of $\sigma\{\mathrm{p}(\sigma)\} = \sigma^+$ satisfying $\mathrm{q}(\mathrm{p}(\sigma), \sigma) \circ \mathrm{v}_\sigma = \mathrm{id}_{\Gamma.\sigma}$. It interprets the last variable in a nonempty context, i.e. $\Gamma, x{:}\sigma \vdash x : \sigma$. The meanings of the other variables are obtained as weakenings of v. The last variable has in particular the expected property $\mathrm{v}_\sigma\{M\} = M$ for $M \in \mathrm{Sect}(\sigma)$.

Interpretation of the syntax. Provided that suitable interpretations of base types and type constructors are given, an a priori partial interpretation function can be defined on pre-terms, -types, and -contexts in such a way that every context is interpreted as a **C**-object, every type is interpreted as an element of Fam at the interpretation of its context and finally terms are interpreted as elements of $\mathrm{Sect}(\sigma)$ where σ is the interpretation of its type. Syntactic context morphisms get interpreted as context morphisms. Moreover, the interpretation is such that definitionally equal objects get interpreted as equal objects. The details of this interpretation function are spelled out in [19] for a slightly different but equivalent class of models and in [17] for cwas.

What it means that a cwa is closed under a type former can be almost directly read off from the syntactic rules. For example, closure under dependent products (Π-types) means the following: for every two families $\sigma \in \mathrm{Fam}(\Gamma)$ and $\tau \in \mathrm{Fam}(\Gamma{\cdot}\sigma)$ there is a family $\Pi(\sigma, \tau) \in \mathrm{Fam}(\Gamma)$; for every section $M \in \mathrm{Sect}(\tau)$ there is a section $\lambda_{\sigma,\tau}(M) \in \mathrm{Sect}(\Pi(\sigma, \tau))$; for every section M of $\Pi(\sigma, \tau)$ and $N \in \mathrm{Sect}(\sigma)$ there is a section $\mathrm{App}_{\sigma,\tau}(M,\ N) \in \mathrm{Sect}(\tau\{N\})$; in such a way that $\mathrm{App}_{\sigma,\tau}(\lambda_{\sigma,\tau}(M), N) = M\{N\}$ and for $f : \mathrm{B} \to \Gamma$ we have $\Pi(\sigma, \tau)\{f\} = \Pi(\sigma\{f\}, \tau\{\mathrm{q}(f, \sigma)\})$ and similar coherence laws for abstraction and application. We follow this pattern when we show that the model we construct does indeed support the required type-formers.

4 Description of the Model

In the following sections we show how types with partial equivalence relations and suitable morphisms define a category with attributes (4.1,4.2 and Prop. 3). Further down, we show that this cwa supports Π-types (4.3) and a universe of propositions with impredicative quantification, i.e. forms a model of the Calculus of Constructions (4.4). We also show that in the model these propositions satisfy principles of proof irrelevance and extensionality (4.5). In Section 5 we then describe how the model supports the quotient types defined in Section 2.

In the model types will be interpreted as types together with Prop-valued partial equivalence relations. Type dependency is modelled only at the level of the relations, i.e. a family indexed over some type σ is a type τ (not depending on σ) and for each $x{:}\sigma$ a partial equivalence relation on τ such that these partial equivalence relations are compatible with the relation on σ.

By analogy to Bishop's definition of sets as assemblies together with an equality relation [4] we call the pairs of types and partial equivalence relations *setoids*.

4.1 Contexts of Setoids

A *context of setoids* is a pair $\Gamma = (\Gamma_{\mathrm{set}}, \Gamma_{\mathrm{rel}})$ where Γ_{set} is a (syntactic) context and Γ_{rel} is a proposition in context $\gamma{:}\,\Gamma_{\mathrm{set}}, \gamma'{:}\,\Gamma_{\mathrm{set}}$, i.e. $\gamma{:}\,\Gamma_{\mathrm{set}}, \gamma'{:}\,\Gamma_{\mathrm{set}} \vdash \Gamma_{\mathrm{rel}}[\gamma, \gamma'] : \mathrm{Prop}$, in such a way that

$$\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma']) \vdash \Gamma_{\mathrm{rel}}[\gamma', \gamma]\ \mathrm{true} \qquad \textsc{Sym}$$

$$\gamma, \gamma', \gamma''{:}\,\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma'])\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma', \gamma'']) \vdash \Gamma_{\mathrm{rel}}[\gamma, \gamma'']\ \mathrm{true} \qquad \textsc{Trans}$$

Here we just write $\mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma'])$ instead of $p : \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma'])$ for some variable p since this variable does not appear in the judgement. We shall adopt this abbrevation in the sequel as well. Two contexts of setoids Γ, Δ are equal if their components are definitionally equal, i.e. if $\vdash \Gamma_{\mathrm{set}} = \Delta_{\mathrm{set}}\ \mathbf{ctxt}$ and $\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}} \vdash \Gamma_{\mathrm{rel}}[\gamma, \gamma'] = \Delta_{\mathrm{rel}}[\gamma, \gamma'] : \mathrm{Prop}$. The implicit witnesses for Sym and Trans are not compared.

The *empty context of setoids* is given by $\mathbf{1} = (\bullet, tt)$ where Sym and Trans are validated by the canonical proof of tt.

Morphisms. Let Γ, Δ be contexts of setoids. A *morphism* from Γ to Δ is a (syntactic) context morphism f from Γ_{set} to Δ_{set} with

$$\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma']) \vdash \Delta_{\mathrm{set}}[f[\gamma], f[\gamma']]\ \mathrm{true} \qquad \textsc{Resp}$$

Two morphisms are equal if their components are definitionally equal. So a morphism must respect the relations associated to its domain and codomain. Clearly, the composition of two morphisms is a morphism again and the identity is a morphism. Moreover, the unique context morphism into $\mathbf{1}$ trivially satisfies Resp so that we have

Proposition 2 *The contexts of setoids and their morphisms form a category with terminal object.*

Notice that we do not identify "provably equal" morphisms, i.e. f and g with $\gamma{:}\,\Gamma_{\mathrm{set}}, \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma]) \vdash \Delta_{\mathrm{rel}}[f[\gamma], g[\gamma']]$ true. Doing so would render the semantic equality undecidable and therefore the source type theory would be undecidable. The difference between actual semantic equality and provable equality is also important for the choice operator we consider in 5.2 which would be unsound if the two were identified.

4.2 Families of Setoids

Let Γ be a context of setoids. A family of setoids indexed over Γ is a pair $\sigma = (\sigma_{\mathrm{set}}, \sigma_{\mathrm{rel}})$ where σ_{set} is a type *in the empty context* ($\bullet \vdash \sigma\ \mathbf{type}$) and

$$\gamma{:}\,\Gamma_{\mathrm{set}}\ ,\ s, s'{:}\,\sigma_{\mathrm{set}} \vdash \sigma_{\mathrm{rel}}[\gamma, s, s'] : \mathrm{Prop}$$

is a family of relations on σ_{set} indexed by Γ_{set} such that

$$\gamma{:}\Gamma_{\mathrm{set}}\ ,\ s,s'{:}\sigma_{\mathrm{set}}\ , \qquad \textsc{Sym}$$
$$\mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma])\ ,\ \mathrm{Prf}(\sigma_{\mathrm{rel}}[\gamma,s,s'])\vdash \sigma_{\mathrm{rel}}[\gamma,s',s]\ \mathrm{true}$$
$$\gamma{:}\Gamma_{\mathrm{set}}\ ,\ s,s',s''{:}\sigma_{\mathrm{set}}\ , \qquad \textsc{Trans}$$
$$\mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma])\ ,\ \mathrm{Prf}(\sigma_{\mathrm{rel}}[\gamma,s,s'])\ ,\ \mathrm{Prf}(\sigma_{\mathrm{rel}}[\gamma,s',s''])\vdash \sigma_{\mathrm{rel}}[\gamma,s,s'']\ \mathrm{true}$$
$$\gamma,\gamma'{:}\Gamma_{\mathrm{set}}\ ,\ s,s'{:}\sigma_{\mathrm{set}}\ , \qquad \textsc{Comp}$$
$$\mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma'])\ ,\ \mathrm{Prf}(\sigma_{\mathrm{rel}}[\gamma,s,s'])\vdash \sigma_{\mathrm{rel}}[\gamma',s,s']\ \mathrm{true}$$

Two families are equal if their two components $-_{\mathrm{set}}$ and $-_{\mathrm{rel}}$ are definitionally equal. Thus, a family of setoids consists of a Γ_{set}-indexed family of partial equivalence relations on one and the same type σ_{set} with the property that the relations over related elements in Γ_{set} are equivalent (by virtue of Comp and symmetry of Γ_{rel}). It is important that symmetry and transitivity are relativised to "existing" $\gamma{:}\Gamma_{\mathrm{set}}$, i.e. those with $\Gamma_{\mathrm{rel}}[\gamma,\gamma]$ since otherwise most of the type formers including quotient types would not go through. Also intuitively this restriction makes sense since σ_{set} and σ_{rel} are meant to be undefined or unconstrained outside the "existing" part of Γ. The set of families of setoids over Γ is denoted by $\mathrm{Fam}(\Gamma)$.

Substitution and comprehension. Let B,Γ be contexts of setoids, $f:\mathrm{B}\to\Gamma$ and $\sigma\in\mathrm{Fam}(\Gamma)$. A family $\sigma\{f\}\in\mathrm{Fam}(\mathrm{B})$ is defined by $\sigma\{f\}_{\mathrm{set}}=\sigma_{\mathrm{set}}$ and $\sigma\{f\}_{\mathrm{rel}}[\beta{:}\mathrm{B}_{\mathrm{set}}\ ,\ s,s'{:}\sigma_{\mathrm{set}}]=\sigma_{\mathrm{rel}}[f[\beta],s,s']$. The comprehension of σ is defined by $(\Gamma\cdot\sigma)_{\mathrm{set}}=\Gamma_{\mathrm{set}},s{:}\sigma_{\mathrm{set}}$ and $(\Gamma\cdot\sigma)_{\mathrm{rel}}[(\gamma,s)\ ,\ (\gamma',s')]=\Gamma_{\mathrm{rel}}[\gamma,\gamma']\wedge\sigma_{\mathrm{rel}}[\gamma,s,s']$.

The morphism $\mathrm{p}(\sigma):\Gamma\cdot\sigma\to\Gamma$ is defined by $\mathrm{p}(\sigma)[\gamma,s]=\gamma$; the relation is preserved by $\wedge$-elimination. Finally, the morphism $\mathrm{q}(f,\sigma):\mathrm{B}\cdot\sigma\{f\}\to\Gamma\cdot\sigma$ is given by $\mathrm{q}(f,\sigma)[\beta,s]=(f[\beta],s)$. This preserves the relation since f does. It is also readily checked that the square (‡‡‡) from Def. 1 is a pullback.

Proposition 3 *Contexts of setoids together with their morphisms and families form a cwa—the* setoid model.

Since it is induced by definitional equality in the target type theory, the semantic equality in the setoid model is obviously decidable. We believe that one can obtain a model of extensional type theory (where propositional and definitional equality coincide) by identifying "related" morphisms and contexts and families with equivalent relations. In this model semantic equality would be undecidable and the choice operator (see 5.2) would be unsound.

Sections. Let $\sigma\in\mathrm{Fam}(\Gamma)$. A section of σ is a syntactic context morphism $M:\Gamma_{\mathrm{set}}\Rightarrow\Gamma_{\mathrm{set}},x{:}\sigma_{\mathrm{set}}$ which is the identity on the variables in Γ. Therefore, such a section is entirely determined by its last component. It is appropriate to identify sections with these last components so that henceforth a section of σ will be a term $\Gamma_{\mathrm{set}}\vdash M:\sigma_{\mathrm{set}}$ which respects the relations, i.e.

$$\gamma,\gamma'{:}\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma'])\vdash\sigma_{\mathrm{rel}}[\gamma,M[\gamma],M[\gamma']]\ \mathrm{true} \qquad \textsc{Resp}$$

4.3 Dependent Products

Let $\sigma \in \mathrm{Fam}(\Gamma)$ and $\tau \in \mathrm{Fam}(\Gamma \cdot \sigma)$. The dependent product $\Pi(\sigma,\tau)$ is defined by $\Pi(\sigma,\tau)_{\mathrm{set}} = \sigma_{\mathrm{set}} \to \tau_{\mathrm{set}}$ and $\Pi(\sigma,\tau)_{\mathrm{rel}}[\gamma{:}\Gamma_{\mathrm{set}}\ ,\ u,v{:}\Pi(\sigma,\tau)_{\mathrm{set}}] = \forall s,s'{:}\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma,s,s'] \Rightarrow \tau_{\mathrm{rel}}[(\gamma,s)\ ,\ u\ s\ ,\ v\ s']$. If $M \in \mathrm{Sect}(\tau)$ then we define $\lambda_{\sigma,\tau}(M)[\gamma{:}\Gamma_{\mathrm{set}}] = \lambda s{:}\sigma_{\mathrm{set}}.M[\gamma,s]$ and conversely if $M \in \mathrm{Sect}(\Pi(\sigma,\tau))$ and $N \in \mathrm{Sect}(\sigma)$ we define $\mathrm{App}_{\sigma\ ,\tau}(M,N)[\gamma{:}\Gamma_{\mathrm{set}}] = (M[\gamma]\ N[\gamma])$ Now by straightforward calculation we obtain:

Proposition 4 *These data endow the setoid model with dependent products.*

In very much the same way we can interpret Σ-types, natural numbers, and various inductive types present in the target type theory. Since the interpretation of these is almost forced we leave it out here.

4.4 Propositions

Our next goal is to identify a type of propositions (and proofs) which allows us to "externalise" the relations associated to each type and to interpret quotient types and extensionality. The semantic version of propositions does not exactly match the syntactic rules, for we make use of the following abstract formulation.

Proposition 5 *Assume some cwa with Π-types. It admits the interpretation of the Calculus of Constructions if the following are given.*

- *distinguished families* $\mathbf{Prop} \in \mathrm{Fam}(\mathbf{1})$ *and* $\mathbf{Prf} \in \mathrm{Fam}(\mathbf{1}\cdot\mathbf{Prop})$
- *for each* $\sigma \in \mathrm{Fam}(\Gamma)$ *and* $S : \Gamma\cdot\sigma \to \mathbf{1}\cdot\mathbf{Prop}$[3] *a distinguished morphism* $\forall_\sigma(S) : \Gamma \to \mathbf{1}\cdot\mathbf{Prop}$ *and a distinguished morphism* $\mathrm{ev}_{\sigma,S} : \Gamma\cdot\sigma\cdot\mathbf{Prf}\{\forall_\sigma(S)\circ \mathrm{p}(\sigma)\} \to \Gamma\cdot\sigma\cdot\mathbf{Prf}\{S\}$ *with* $\mathrm{p}(\mathbf{Prf}\{S\})\circ \mathrm{ev}_{\sigma,S} = \mathrm{p}(\mathbf{Prf}\{S\})$
- *for each section* $M \in \mathrm{Sect}(\mathbf{Prf}\{S\})$ *a distinguished section* $\lambda_{\sigma,S}(M) \in \mathrm{Sect}(\mathbf{Prf}\{\forall_\sigma(S)\})$ *with* $\mathrm{ev}_{\sigma,S}\circ\lambda_{\sigma,S}(M)\{\mathrm{p}(\sigma)\} = M$

in such a way that all these data are stable under substitution, that is for $f : \mathrm{B} \to \Gamma$, $\sigma \in \mathrm{Fam}(\Gamma)$, $S : \Gamma\cdot\sigma \to \mathbf{1}\cdot\mathbf{Prop}$, $M \in \mathrm{Sect}(\mathbf{Prf}\{S\})$ *we have* $\forall_\sigma(S)\circ f = \forall_{\sigma\{f\}}(s\circ f^+)$ *and* $\mathrm{ev}_{\sigma,S}\circ f^{++} = f^{++}\circ \mathrm{ev}_{\sigma\{f\},s\circ f^+}$ *and* $\lambda_{\sigma,S}(M)\{f\} = \lambda_{\sigma\{f\},s\circ f^+}(M\{f^+\})$

Proof (Sketch). One interprets types of the form $\mathrm{Prf}(P)$ as morphisms into $\mathbf{1}\cdot\mathbf{Prop}$. The dependent product $\Pi x{:}\sigma.\tau$ gets interpreted by the semantic dependent product if τ is not of the form $\mathrm{Prf}(-)$ and by $\forall$ otherwise. Accordingly for application and abstraction.

In the sequel we denote by "**Prop**" both the family in Fam(1), and its comprehension $\mathbf{1}\cdot\mathbf{Prop}$. To simplify the exposition we assume that the target type theory supports an extensional unit type, i.e. there is a type $\Gamma \vdash 1\ \mathbf{type}$ in every context, and a term $\Gamma \vdash \star : 1$ together with the equation $\Gamma \vdash M = \star : 1$ for every $\Gamma \vdash M : 1$. We use the extensional unit type as the underlying type

[3] Upper case is used to avoid confusion with the variable s.

of propositional types. It is possible to get rid of the extensional unit type, by defining families of setoids as being either of the form defined in Section 4.2, or to consist simply of a proposition in context Γ_{set} compatible with Γ_{rel} (in which case it is silently understood that the $-_{\mathrm{set}}$-component is the extensional unit type). Comprehension, substitution, and the type formers have then to be defined by case distinction.

We are ready now ready to define the required ingredients to interpret propositions. The underlying type of $\mathbf{Prop} \in \mathrm{Fam}(\mathbf{1})$ in the setoid model is just the syntactic type of propositions.

$$\mathbf{Prop}_{\mathrm{set}} = \mathrm{Prop}$$

In order to identify equivalent propositions we define the relation as bi-implication.

$$\mathbf{Prop}_{\mathrm{rel}}[p, q{:}\,\mathrm{Prop}] = p \Leftrightarrow q$$

This is clearly symmetric and transitive and so a family over $\mathbf{1}$ has been defined. Note that Comp degenerates to a tautology in the case of families over $\mathbf{1}$.

The family $\mathbf{Prf} \in \mathrm{Fam}(\mathbf{1} \cdot \mathbf{Prop})$ is given by

$$\begin{aligned}\mathbf{Prf}_{\mathrm{set}}[p{:}\,\mathrm{Prop}] &= 1\\ \mathbf{Prf}_{\mathrm{rel}}[p{:}\,\mathrm{Prop}\ ,\ x, x'{:}\,1] &= p\end{aligned}$$

where 1 is the extensional unit type. The relation on $\mathbf{Prf}$ is trivially symmetric and transitive; for Comp we use the relation on $\mathbf{Prop}$. More precisely, if $\mathbf{Prop}_{\mathrm{rel}}[p, q]$ and $\mathbf{Prf}_{\mathrm{rel}}[p, x, y]$, i.e. p, then also q, thus $\mathbf{Prf}_{\mathrm{rel}}[q, x, y]$.

Next we define universal quantification. If $\sigma \in \mathrm{Fam}(\Gamma)$ and $S : \Gamma \cdot \sigma \to \mathbf{Prop}$ then we put

$$\forall_\sigma(S)[\gamma{:}\,\Gamma_{\mathrm{set}}] = \forall x{:}\,\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma, s, s] \Rightarrow S[\gamma, x]$$

For property Resp assume $\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}$ and $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$. We must show that $\forall_\sigma(S)[\gamma] \Leftrightarrow \forall_\sigma(S)[\gamma']$. So assume $\forall_\sigma(S)[\gamma]$ and $x{:}\,\sigma_{\mathrm{set}}$ with $\sigma_{\mathrm{rel}}[\gamma', x, x]$. Using Sym and Comp we get $\sigma_{\mathrm{rel}}[\gamma, x, x]$, hence $S[\gamma, x]$ by assumption. Property Resp for S using $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$ and $\sigma_{\mathrm{rel}}[\gamma, x, x]$ gives $S[\gamma, x] \Leftrightarrow S[\gamma', x]$ and thus $S[\gamma', x]$ as required. The other direction is symmetric. We see that the relativisation to "existing" $x{:}\,\sigma_{\mathrm{set}}$ is necessary to prove Resp.

For the abstraction let $M \in \mathrm{Sect}(\mathbf{Prf}\{S\})$. We define $\lambda_{\sigma,S}(M)[\gamma{:}\,\Gamma_{\mathrm{set}}] = \star$. To see that this is indeed a section of $\mathbf{Prf}\{\forall_\sigma(S)\}$ assume $\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}$ and $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$. We must show that

$$\mathbf{Prf}\{\forall_\sigma(S)\}_{\mathrm{rel}}[\gamma, \lambda_{\sigma,S}(M)[\gamma]\ ,\ \lambda_{\sigma,S}(M)[\gamma']]$$

which equals $\forall x{:}\,\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma, x, x] \Rightarrow S[\gamma, x]$ by definition of $\mathbf{Prf}_{\mathrm{rel}}$ and $\forall$. Now assuming $x{:}\,\sigma_{\mathrm{set}}$ and $\sigma_{\mathrm{rel}}[\gamma, x, x]$ we have that $\Gamma \cdot \sigma_{\mathrm{rel}}[(\gamma, x)\ ,\ (\gamma', x)]$, hence $S[\gamma, x]$ by Resp for M.

Finally, the evaluation morphism is given by $\mathrm{ev}_{\sigma,S}[\gamma{:}\,\Gamma_{\mathrm{set}}\ ,\ x{:}\,\sigma_{\mathrm{set}}\ ,\ u{:}\,1] = (\gamma, x, \star)$. We must show that this defines a morphism from $\Gamma \cdot \sigma \cdot \mathbf{Prf}\{\forall_\sigma(S) \circ \mathrm{p}(\sigma)\}$ to $\Gamma \cdot \sigma \cdot \mathbf{Prf}\{S\}$. Assume $\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}$, $x, x'{:}\,\sigma_{\mathrm{set}}$ with $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$ and $\sigma_{\mathrm{rel}}[\gamma, x, x']$;

furthermore assume $\forall x{:}\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma, x, x] \Rightarrow S[\gamma, x]$ (meaning that the two variables of unit type corresponding to $\mathbf{Prf}\{\forall_\sigma(S) \circ \mathrm{p}(\sigma)\}$ are "related" according to the definition of $\mathbf{Prf}_{\mathrm{rel}}$. We must show that $S[\gamma, x]$. But this follows since by SYM and TRANS for Γ we have $\Gamma_{\mathrm{rel}}[\gamma, \gamma]$ and thus $\sigma_{\mathrm{rel}}[\gamma, x, x]$ by SYM and TRANS for σ.

The equations relating evaluation and abstraction follow straightforwardly from the properties of the extensional unit type. Also the stability under substitution of all components follows readily by expanding the definitions.

Proposition 6 *The setoid model admits the interpretation of the Calculus of Constructions.*

4.5 Proof Irrelevance and Extensionality

Next we are going to explore which additional propositions are provable in the setoid model. First, we have a non-provability result:

Proposition 7 (Consistency) *The family* $\mathit{ff} := \mathbf{Prf}\{\forall_{\mathbf{Prop}}(\mathrm{id}_{\mathbf{Prop}})\} \in \mathrm{Fam}(\mathbf{1})$ *has no sections.*

Proof. We have $\mathit{ff}_{\mathrm{set}} = 1$ and $\mathit{ff}_{\mathrm{rel}}[x, y{:}\,1] = \forall p{:}\,\mathrm{Prop}.\mathbf{Prop}_{\mathrm{rel}}[p, p] \Rightarrow p$. A section of ff consists of an element of 1 (necessarily $\star$) such that $\forall p{:}\,\mathrm{Prop}.(p \Leftrightarrow p) \Rightarrow p$ (expanding $\mathbf{Prop}_{\mathrm{rel}}$). But this is not possible by consistency of the syntax.

Next we look at Leibniz equality in the model. If $\sigma \in \mathrm{Fam}(\Gamma)$ and $M, N \in \mathrm{Sect}(\sigma)$ then let $\mathrm{L_Eq}(M, N) : \Gamma \to \mathbf{Prop}$ be the denotation of Leibniz equality. By unfolding the definition we find that in the setoid model

$$\begin{aligned}\mathrm{L_Eq}(M, N)[\gamma{:}\,\Gamma_{\mathrm{set}}] = \forall P{:}\,\sigma_{\mathrm{set}} \to \mathrm{Prop}.\\ (\forall x, x'{:}\,\sigma.\sigma_{\mathrm{rel}}[\gamma, x, x'] \Rightarrow ((P\ \ x) \Leftrightarrow (P\ \ x'))) \Rightarrow\\ \forall y{:}\,1.(P\ \ M[\gamma]) \Rightarrow (P\ \ N[\gamma])\end{aligned}$$

So (modulo the trivial quantification over 1) M and N are Leibniz equal in the model if they are indistinguishable by observations *which respect the relation on* σ.

Lemma 8 *For* $\sigma \in \mathrm{Fam}(\Gamma)$ *and* $M, N \in \mathrm{Sect}(\sigma)$ *the set* $\mathrm{Sect}(\mathbf{Prf}\{\mathrm{L_Eq}(M, N)\})$ *is nonempty iff* $\gamma{:}\,\Gamma_{\mathrm{set}}$, $\mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma]) \vdash \sigma_{\mathrm{rel}}[\gamma, M[\gamma], N[\gamma]]$ true

Proof. Assume a section of $\mathbf{Prf}\{\mathrm{L_Eq}(M, N)\}$. By definition this means that if $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$ for some $\gamma, \gamma'{:}\,\Gamma_{\mathrm{set}}$ then $M[\gamma]$ and $N[\gamma]$ (not $N[\gamma']$!) are indistinguishable by observations respecting σ_{rel}. Now $\lambda x{:}\,\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma, M[\gamma], x]$ is such an observation as can be seen using SYM and TRANS for σ and $\Gamma_{\mathrm{rel}}[\gamma, \gamma]$ by virtue of $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$ and SYM, TRANS for Γ. Therefore we can deduce $\sigma_{\mathrm{rel}}[\gamma,\ M[\gamma], N[\gamma]]$ provided we can show $\sigma_{\mathrm{rel}}[\gamma, M[\gamma], M[\gamma]]$, but this follows from RESP for M. Conversely, if $\sigma[\gamma, M[\gamma], N[\gamma]]$ and $P : \sigma_{\mathrm{set}} \to \mathrm{Prop}$ respects σ_{rel} then obviously $(P\ \ M[\gamma]) \Rightarrow (P\ \ N[\gamma])$ by definition of "respects".

Thus Leibniz equality "externalises" the relation associated to each family.

Proposition 9 *The following rules can be interpreted in the setoid model.*

$$\frac{\Gamma \vdash A : \mathrm{Prop} \qquad \Gamma \vdash M, N : \mathrm{Prf}(A)}{\Gamma \vdash M = N : \mathrm{Prf}(A)} \ \textsc{Pr-Ir}$$

$$\frac{\Gamma \vdash P : \mathrm{Prop} \qquad \Gamma \vdash Q : \mathrm{Prop} \qquad \Gamma \vdash H : \mathrm{Prf}(P \Leftrightarrow Q)}{\Gamma \vdash \mathrm{Bi_Imp}(P, Q, H) : \mathrm{Prf}(P \stackrel{L}{=} Q)} \ \textsc{Bi-Imp}$$

$$\frac{\Gamma \vdash U, V : \Pi x{:}\sigma.\tau \qquad \Gamma, x{:}\sigma \vdash H : \mathrm{Prf}(U\ x \stackrel{L}{=} V\ x)}{\Gamma \vdash \mathrm{Ext}(H) : U \stackrel{L}{=} V} \ \textsc{Ext}$$

Proof. We identify syntactic objects with their denotations in the setoid model. Rule Pr-Ir is immediate because both M and N equal $\star$. For rule Bi-Imp assume $P, Q \in \mathrm{Sect}(\mathbf{Prop}\{!_\Gamma\})$. The assumption H gives rise to a proof of $P[\gamma] \Leftrightarrow Q[\gamma]$ for every $\gamma : \Gamma_{\mathrm{set}}$ with $\Gamma_{\mathrm{rel}}[\gamma, \gamma]$. But by Lemma 8 this implies that there exists a section of $\mathbf{Prf}\{\mathrm{L_Eq}(P, Q)\}$. The proof for Ext is similar.

Now we shall present a somewhat unexpected feature of Leibniz equality, namely that it behaves like Martin-Löf's identity type in the sense that a Leibniz principle for dependent types can be interpreted from which the definability of Martin-Löf's elimination rule [15] follows using Pr-Ir (see e.g. [8]).

Proposition 10 *For each* $\sigma \in \mathrm{Fam}(\Gamma)$ *and* $\tau \in \mathrm{Fam}(\Gamma \cdot \sigma)$ *and* $M, N \in \mathrm{Sect}(\sigma)$ *and* $P \in \mathrm{L_Eq}(M, N)$ *and* $U \in \mathrm{Sect}(\tau\{M\})$ *there exists a well-determined section* $\mathrm{Subst}_{\sigma,\tau}(P, U) \in \mathrm{Sect}(\tau\{N\})$ *in such a way that* $\mathrm{Subst}_{\sigma,\tau}(\mathrm{Refl}(M), U) = U$, *where* $\mathrm{Refl}(M)$ *is the canonical section of* $\mathrm{L_Eq}(M, M)$ *corresponding to reflexivity. Moreover, this operator* Subst *is stable under substitution in all its arguments.*

Proof. We define $\mathrm{Subst}_{\sigma,\tau}(P, U)$ simply as U. That this is a section of $\tau\{N\}$ follows from Lemma 8 applied to P and rule Comp for the family τ. The other properties are trivially satisfied.

5 Quotient types

We now turn to the interpretation of quotient types in the model. Recall their defining rules from Section 2 . Let $\sigma \in \mathrm{Fam}(\Gamma)$ and $R : \Gamma \cdot \sigma \cdot \sigma^+ \to \mathbf{Prop}$ be a "relation" over σ. Viewed internally R is a term of type $\gamma{:}\Gamma_{\mathrm{set}}\ ,\ x, x'{:}\sigma_{\mathrm{set}} \vdash \mathrm{Prop}$ **type** such that

$$\begin{array}{l}\gamma{:}\Gamma_{\mathrm{set}}\ ,\ x, y{:}\sigma_{\mathrm{set}}\ ,\ \gamma'{:}\Gamma_{\mathrm{set}}\ ,\ x', y'{:}\sigma_{\mathrm{set}}\ , \\ \qquad \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma, \gamma'] \wedge \sigma_{\mathrm{rel}}[\gamma, x, x'] \wedge \sigma_{\mathrm{rel}}[\gamma, y, y']) \vdash \\ \qquad\qquad R[x, y] \Leftrightarrow R[x', y']\ \mathrm{true}\end{array}$$

We say that R "respects" σ_{rel}. Our aim is to define a new family σ/R on which the relation is given by R. The underlying type remains unchanged by quotienting: $(\sigma/R)_{\mathrm{set}}[\gamma] = \sigma_{\mathrm{set}}[\gamma]$. Now since R is not guaranteed to be symmetric and transitive, we have to take the symmetric and transitive closure of R, moreover we must ensure compatibility with σ_{rel} — the relation already present on σ. It

turns out that the right choice for $(\sigma/R)_{\mathrm{rel}}$ is the following higher-order encoding of symmetric, transitive closure.

$$\begin{aligned}
&(\sigma/R)_{\mathrm{rel}}[\gamma{:}\Gamma_{\mathrm{set}}\ ,\ s,s'{:}\sigma_{\mathrm{set}}] = \forall R'{:}\sigma_{\mathrm{set}} \to \sigma_{\mathrm{set}} \to \mathrm{Prop}.\\
&\qquad (\forall x,x'{:}\sigma_{\mathrm{set}}.(R'\ x\ x') \Rightarrow (R'\ x'\ x)) && (\boldsymbol{\sigma})\\
&\quad \Rightarrow (\forall x,x',x''{:}\sigma_{\mathrm{set}}.(R'\ x\ x') \Rightarrow (R'\ x'\ x'') \Rightarrow (R'\ x\ x'')) && (\boldsymbol{\tau})\\
&\quad \Rightarrow (\forall x,x'{:}\sigma_{\mathrm{set}}.\sigma_{\mathrm{rel}}[\gamma,x,x'] \Rightarrow (R'\ x\ x')) && (\boldsymbol{\rho})\\
&\quad \Rightarrow (\forall x,x'{:}\sigma_{\mathrm{set}}.R[\gamma,x,x'] \Rightarrow \sigma_{\mathrm{rel}}[\gamma,x,x] \Rightarrow \sigma_{\mathrm{rel}}[\gamma,x',x'] \Rightarrow (R'\ x\ x')) && (\boldsymbol{\kappa})\\
&\qquad \Rightarrow (R'\ s\ s')
\end{aligned}$$

In other words, $(\sigma/R)_{\mathrm{rel}}[\gamma,-,-]$ is the least equivalence relation on σ_{set} which contains $\sigma_{\mathrm{rel}}[\gamma,-,-]$ and $R[\gamma,-,-]$ restricted to the domain of $\sigma_{\mathrm{rel}}[\gamma,-,-]$. We shall call such a relation *suitable*, so $(\sigma/R)_{\mathrm{rel}}$ is the least suitable relation.

5.1 Equivalence Classes

Assume $M \in \mathrm{Sect}(\sigma)$. We want to construct a section $[M]_R$ of σ/R. We put $[M]_R[\gamma] = M[\gamma]$ so $[M]_R$ behaves just like M. We must prove Resp for $[M]_R$. Let $\gamma,\gamma'{:}\Gamma_{\mathrm{set}}$ and $\Gamma_{\mathrm{rel}}[\gamma,\gamma']$. Moreover, let $R'{:}\sigma_{\mathrm{set}} \to \sigma_{\mathrm{set}} \to \mathrm{Prop}$ be a suitable relation. We must show that $R'[M[\gamma],M[\gamma']]$. Since M is a section we have $\sigma_{\mathrm{rel}}[\gamma,M[\gamma],M[\gamma']]$ so we are done since suitable relations contain $\sigma_{\mathrm{rel}}[\gamma,-,-]$.

For Q-Ax assume $M,N \in \mathrm{Sect}(\sigma)$ and $H \in \mathrm{Sect}(\mathrm{Prf}\{R \circ M^+ \circ N\})$, i.e.

$$\gamma,\gamma'{:}\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma']) \vdash R[\gamma,M[\gamma],N[\gamma]]\ \mathrm{true}$$

We want to show that $\mathrm{Sect}(\mathrm{Prf}\{\mathrm{L_Eq}([M]_R,[N]_R)\})$ is nonempty. By Lemma 8 and the definition of $[-]_R$ it suffices to show that

$$\gamma{:}\Gamma_{\mathrm{set}}\ ,\ \mathrm{Prf}(\Gamma_{\mathrm{rel}}[\gamma,\gamma]) \vdash \sigma/R_{\mathrm{rel}}[\gamma,M[\gamma],N[\gamma]]\ \mathrm{true}$$

In this context we can show $\sigma_{\mathrm{rel}}[\gamma,M[\gamma],M[\gamma]]$ and $\sigma_{\mathrm{rel}}[\gamma,N[\gamma],N[\gamma]]$ using Resp. Thus for every suitable relation $R'{:}\sigma_{\mathrm{set}} \to \sigma_{\mathrm{set}} \to \mathrm{Prop}$ we have $(R'\ M[\gamma]\ N[\gamma])$ by $(\boldsymbol{\kappa})$ which gives the desired result. Similarly, we can define lifting (Q-Elim) and induction (Q-Ind) so that we conclude

Proposition 11 *The setoid model allows the interpretation of quotient types.*

Effectiveness of quotient types. Here we want to address the question as to whether the converse to Q-Ax is also true, i.e. whether we can conclude $R[M,N]$ from $[M]_R \overset{L}{=} [N]_R$. In general, this cannot be the case because together with Q-Ax this implies that R is an equivalence relation. Quotient types are called *effective* [11, 14] if this condition is already sufficient. It turns out that in the setoid model all quotient types are effective and that this is a purely syntactic consequence of the rules for quotient types and rule Bi-Imp.

Proposition 12 *Fix a type theory with quotient types and* Bi-Imp. *Let* $\Gamma \vdash \sigma$ *and* $\Gamma\ ,\ x,y{:}\sigma \vdash R[x,y] : \mathrm{Prop}$ *be an equivalence relation, i.e. in context* Γ *we have* $\forall x{:}\sigma.R[x,x]$ true, *and* $\forall x,y{:}\sigma.R[x,y] \Rightarrow R[y,x]$ true, *and* $\forall x,y,z{:}\sigma.R[x,y] \Rightarrow R[y,z] \Rightarrow R[x,z]$ true. *Then for each* $\Gamma \vdash U,V{:}\sigma$ *with* $[U]_R \overset{L}{=} [V]_R$ *we have* $R[U,V]$.

Proof. Consider the term $M[y{:}\sigma] = R[U,y]$: Prop. We have Γ , $y,y'{:}\sigma$, $R[y,y'] \vdash M[y] \stackrel{L}{=} M[y']$ by symmetry and transitivity and Bi-Imp. Let H denote the proof of this. Now put $M'[z{:}\sigma/R] = \mathrm{plug}_R\, z \,\mathrm{in}\, M \,\mathrm{using}\, H$: Prop. By Q-Comp $M'[[U]_R]$ equals $R[U,U]$ and $M'[[V]_R]$ equals $R[U,V]$. Now the former is true by reflexivity of R and both are Leibniz equal by assumption on U and V. Therefore $R[U,V]$ is true as well.

5.2 A Choice Operator for Quotient Types

Our goal consists of finding a way of getting hold of a representative for a given element of a quotient type. The idea is that since in the model an element M of a quotient type is nothing but an element of the underlying type, albeit with a weaker Resp requirement, it should under certain circumstances be possible to view M as an element of this underlying type. This is formalised by the rules Q-Choice, Q-Choice-Ax and Q-Choice-Comp from Section 2 which we do not reproduce here for lack of space.

Semantically, we want to interpret the choice operator as the identity, i.e. if $M \in \mathrm{Sect}(\sigma/R)$ then we want to put $\mathrm{choice}(M)[\gamma{:}\,\Gamma_{\mathrm{set}}] = M[\gamma]$. Now from $M \in \mathrm{Sect}(\sigma/R)$ we can deduce that if $\Gamma_{\mathrm{rel}}[\gamma,\gamma']$ for some $\gamma,\gamma'{:}\,\Gamma_{\mathrm{set}}$ then $\sigma/R_{\mathrm{rel}}[\gamma, M[\gamma]$, $M[\gamma']]$. But in order to conclude $\mathrm{choice}(M) = M \in \mathrm{Sect}(\sigma)$ we need to have (the stronger) $\sigma_{\mathrm{rel}}[\gamma, M[\gamma], M[\gamma']]$. One situation in which we can deduce the latter from the former occurs when from $\Gamma_{\mathrm{rel}}[\gamma,\gamma']$ we can conclude that $M[\gamma]$ and $M[\gamma']$ are actually Leibniz equal. Now this situation in turn occurs for example when $\Gamma_{\mathrm{rel}}[\gamma,\gamma']$ entails that γ and γ' are Leibniz equal themselves. This motivates the following definition.

Definition 13 *The set of* non-quotiented *types is defined by the following clauses.*

- Nat *(and other inductive types) and* $\Pi x_1{:}\sigma_1 \ldots \Pi x_n{:}\sigma_n.\mathrm{Prf}(M)$ *for* $n \geq 0$ *are non-quotiented.*
- *If* σ *and* τ *are non-quotiented, so is* $\Sigma x{:}\sigma.\tau$.

A (syntactic) context Γ *is non-quotiented if it is made up out of non-quotiented types only.*

Notice that if $\vdash \Gamma = \Delta$ and Γ is non-quotiented then so is Δ. An easy induction on the definition of "non-quotiented" gives the following semantic counterpart.

Proposition 14 *Let* Γ *be a non-quotiented syntactic context and* $(\Gamma_{\mathrm{set}}, \Gamma_{\mathrm{rel}})$ *be its interpretation in the setoid model.* $\gamma,\gamma'{:}\,\Gamma_{\mathrm{set}}$, $\Gamma_{\mathrm{rel}}[\gamma,\gamma'] \vdash \gamma.i \stackrel{L}{=} \gamma'.i$ true *for all* $i \leq$ *the length of* Γ.

Now we could semantically justify the application of choice in non-quotiented contexts, i.e. the above "certain proviso" would be that Γ is non-quotiented. With such a rule we would, however lose the syntactic weakening and substitution properties which are implicit in e.g. the typing rule for application. For example if $\vdash M : \sigma/R$ we can infer $\vdash \mathrm{choice}(M) : \sigma$ because the empty context is non-quotiented, but we would not have $x{:}\sigma/R \vdash \mathrm{choice}(M) : \sigma$ although x does

not occur in M. Therefore, we close the rule up under arbitrary substitution and weakening and thus arrive at the following definitive rule for choice:

$$\frac{\Gamma \vdash M : \sigma/R \qquad \begin{array}{l}\text{There exists a non-quotiented context } \Delta \text{ and}\\ \text{a type } \tau \text{ and a term } N \text{ with } \Delta \vdash N : \tau \text{ and a}\\ \text{syntactic context morphism } f : \Gamma \Rightarrow \Delta \text{ such}\\ \text{that } M \equiv N[f] \text{ and } \sigma/R \equiv \tau[f].\end{array}}{\Gamma \vdash \mathrm{choice}(M) : \sigma} \quad \text{Q-Choice}$$

Here $\equiv$ means syntactic identity and not just definitional equality so that the side condition is decidable since the possible substitutions f are bounded by the size of M. The condition is e.g. satisfied for $z{:}\mathrm{Int} \vdash +|z| : \mathrm{Int}$ with $\Delta = \mathrm{Nat}$ and $f[z{:}\mathrm{Int}] = |z|$, but it does not hold for $z{:}\mathrm{Int} \vdash z + z : \mathrm{Int}$

Proposition 15 *The above rules* Q-Choice, Q-Choice-Comp, *and* Q-Choice-Ax *can be soundly interpreted in the setoid model.*

Proof. We interpret $\mathrm{choice}(M)$ like M. Suppose that $\Gamma \vdash \mathrm{choice}(M) : \sigma$ by Q-Choice then $M \equiv N[f]$ for some (syntactic) substitution $f : \Gamma \Rightarrow \Delta$. We identify these syntactic objects with their interpretations. If $\Gamma_{\mathrm{rel}}[\gamma, \gamma']$ then $f[\gamma]$ and $f[\gamma']$ are actually Leibniz equal by Prop. 14 using the assumption that Δ is non-quotiented. So $M[\gamma]$ and $M[\gamma']$ are Leibniz equal and thus related in σ_{rel}. Thus we can assert $M \in \mathrm{Sect}(\sigma)$. Since both choice and $[-]_R$ are interpreted as the identity the rules Q-Choice-Comp and Q-Choice-Ax are validated.

Discussion. The choice operator is quite unusual and seems paradoxical at first so that a few examples explaining its use are in order. Let $-1 := [(0,1)]_{R_{\mathrm{Int}}}$ and $-1' := [(2,3)]_{R_{\mathrm{Int}}}$ where $1, 2, 3$ are abbreviations for the corresponding numerals of type Nat. Now since $0+3 \stackrel{L}{=} 1+2$ we have $R_{\mathrm{Int}}[(0,1)\ ,(2,3)]$, and thus $-1 \stackrel{L}{=} -1'$. On the other hand $\mathrm{choice}(-1) = (0,1)$ and $\mathrm{choice}(-1') = (2,3)$. It seems as if we could conclude $(0,1) \stackrel{L}{=} (2,3)$ which would be a contradiction. Now $-1 \stackrel{L}{=} -1'$ means $\forall P{:}\mathrm{Int} \to \mathrm{Prop}.(P\ -1) \Rightarrow (P\ -1')$. In order to get a contradiction from this, we would have to instantiate with $P = \lambda z{:}\mathrm{Int}.\mathrm{choice}(-1) \stackrel{L}{=} \mathrm{choice}(z)$. But this expression is not well-typed since $z{:}\mathrm{Int} \vdash \mathrm{choice}(z)$ (the context $z{:}\mathrm{Int}$ contains a quotient type.) is not. Thus, in some sense the choice operator does not respect Leibniz equality. As any term former it does of course respect definitional equality so in an extensional setting where the two equalities are identified choice would be unsound.

The main usage of the choice operator is that it permits to recover the underlying implementation of a function between quotients internally. For example if we have defined some function $F : \mathrm{Int} \to \mathrm{Int}$ in non-quotiented context Γ (so in particular F may not just be a variable) then we may form $\Gamma, u{:}\mathrm{Nat} \times \mathrm{Nat} \vdash F\ [u]_{R_{\mathrm{Int}}} : \mathrm{Int}$, apply choice and abstract from u to obtain $F' := \lambda u{:}\mathrm{Nat} \times \mathrm{Nat}.\mathrm{choice}(F\ [u]_{R_{\mathrm{Int}}}) : (\mathrm{Nat} \times \mathrm{Nat}) \to (\mathrm{Nat} \times \mathrm{Nat})$ Now from Q-Choice-Ax we obtain $\Gamma\ , u{:}\mathrm{Nat} \times \mathrm{Nat} \vdash F\ [u]_{R_{\mathrm{Int}}} = [F'\ u]_{R_{\mathrm{Int}}} : \mathrm{Int}$ and moreover—since R_{Int} is an equivalence relation— $\Gamma \vdash \forall u, v{:}\mathrm{Nat} \times \mathrm{Nat}.R_{\mathrm{Int}}[u, v] \Rightarrow R_{\mathrm{Int}}[F'\ u, F'\ v]$ using the above and Prop. 12.

Yet another application of choice arises from the use of quotient types to model "nonconstructive" types like the real numbers. Assume that we have defined a type of real numbers Real as a quotient of a suitable subset of Nat $\to$ Nat (thought of as of decimal or continued fraction expansions) the quotienting relation being that the difference is a fundamental sequence defined in the usual ε-δ style. Now since different real numbers can be arbitrarily close together, there can be no non-constant function from the quotient type Real to a ground type like Nat. This has been brought forward as a serious argument against the use of quotienting in a constructive setting, since if one has defined a real number one cannot even put one's hands on its first digit! Using the choice operator one can at least solve this last problem. Assume that we have defined a real number R in the empty context, say e or π. We may then form $\vdash \mathrm{choice}(R) : \mathrm{Nat} \to \mathrm{Nat}$ and from this extract the desired intensional information like the first digit or other things. However, it is still not possible to write a non-constant *function* from the reals to Nat. Notice that this lack is not particular to the setoid model, but directly inherited from the target type theory which does not provide functions from Nat $\to$ Nat to Nat which respect the "book"-equality for real numbers. Similarly, in the setoid model we have no nonconstant function from Prop to Nat because this is so in the target type theory.

6 Conclusions

As remarked in the Introduction the model does not provide any type dependency other than the one induced by propositions. In particular we have neither universes nor "large eliminations"[1]. Formally, this can be seen as follows. Assume that the target type theory contains an empty type $\mathbf{0}$ and an operator $?_\sigma$ such that $?_\sigma(M) : \sigma$ if $M : \mathbf{0}$. Then in the setoid model we also have this empty type. Now if we had a universe containing the natural numbers and the empty type then we could define a family of types $n{:}\mathrm{Nat} \vdash \sigma[n]$ **type** such that $\sigma[0] = \mathbf{0}$ and $\sigma[\mathrm{Suc}(n)] = \mathrm{Nat}$. Now remember that the $-_{\mathrm{set}}$-component is left unchanged upon substitution. So in view of the first equation σ_{set} there would have to be a function $?_\tau : \sigma_{\mathrm{set}} \to \tau$ for every type τ which contradicts the second equation if $\tau = \mathbf{0}$.

A major application of such families of types is that they allow to derive Peano's fourth axiom [15, 18]. On the level of propositions we are still able to interpret this axiom, that is we have $0 \stackrel{L}{=} \mathrm{Suc}(0) \Rightarrow \mathit{ff}$, provided this holds in the target type theory. The difference is that $\mathrm{Prf}(\mathit{ff})$ is weaker than the empty type because in the presence of an element of $\mathrm{Prf}(\mathit{ff})$ every proposition is true, but not every type is inhabited.

Universes are also used for modularisation and structuring. E.g. using a universe it is possible to define a type of monoids or groups and to identify the construction of the free group over a monoid as a function. We have investigated an extension to the setoid model which admits a universe restricted in the sense that quotient types and inductive types and elimination of them is only allowed within the universe. This is to be reported elsewhere.

The equality in the setoid model is decidable by performing the interpretation and then using a standard decision procedure for definitional equality. One could therefore devise an implementation of the setoid model which would keep track of the interpretations of all the terms and types in the current environment and replace computation on the external terms with computation on these interpretations. We believe that such an implementation would behave even better than an ordinary implementation like Lego because in the model proofs and programs are entirely separated. At this moment no such implementation exists, but we have defined all the semantic operations in Lego and checked the required semantic equations mechanically. In principle this can be used to machine-check derivations in the source type theory in a variable-free combinatory way. But this is of course far too cumbersome for real applications so that a proper implementation of the translation is called for. However, many applications go through without having the additional *definitional* equalities like PR-IR or Q-CHOICE-COMP, and it is enough to have the corresponding Leibniz equalities. One can then work within a Lego context which provides all the rules we have studied as axioms. In this way we have been able to carry out a number of examples including a proof of Tarski's generalised fixpoint theorem (based on a formalisation by Pollack) and a verification of the alternating bit protocol.

References

1. Thorsten Altenkirch. *Constructions, normalization, and inductive types.* PhD thesis, University of Edinburgh, 1994.
2. Hendrik Barendregt. Functional programming and lambda calculus. Handbook of Theoretical Computer Science, Volume B.
3. Michael Beeson. *Foundations of Constructive Mathematics.* Springer, 1985.
4. Errett Bishop and Douglas Bridges. *Constructive Analysis.* Springer, 1985.
5. J. Cartmell. *Generalized algebraic theories and contextual categories.* PhD thesis, Univ. Oxford, 1978.
6. Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.
7. Robert Constable et al. *Implementing Mathematics with the Nuprl Development System.* Prentice-Hall, 1986.
8. Herman Geuvers and Benjamin Werner. On the Church-Rosser property for expressive type systems. LICS '94.
9. M. J. C. Gordon and T. F. Melham. *Introduction to HOL.* Cambridge, 1993.
10. Martin Hofmann. Elimination of extensionality for Martin-Löf type theory. LNCS 806.
11. Bart Jacobs. Quotients in Simple Type Theory. submitted.
12. Bart Jacobs. Comprehension categories and the semantics of type theory. *Theoretical Computer Science*, 107:169–207, 1993.
13. J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic.* Cambridge, 1985.
14. I. Moerdijk and S. Mac Lane. *Sheaves in Geometry and Logic.* Springer, 1992.
15. B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory, An Introduction.* Clarendon Press, Oxford, 1990.
16. Wesley Phoa. An introduction to fibrations, topos theory, the effective topos, and modest sets. Technical Report ECS-LFCS-92-208, LFCS Edinburgh, 1992.

17. Andrew Pitts. Categorical logic. In *Handbook of Theoretical Computer Science*. Elsevier Science, 199? to appear.
18. Jan Smith. The independence of Peano's fourth axiom from Martin-Löf's type theory without universes. *Journal of Symbolic Logic*, 53(3), 1988.
19. Thomas Streicher. *Semantics of Type Theory*. Birkhäuser, 1991.

A Syntax of the target type theory

The sets of *pre-contexts* (Γ), *pre-types* (σ, τ), and *pre-terms* (M, N, L, O, P) are defined by the following grammar. $\Gamma ::= \bullet | \Gamma, x{:}\sigma$ and $\sigma, \tau ::= 1 | \mathrm{Nat} | \Pi x{:}\sigma.\tau | \Sigma x{:}\sigma.\tau | \mathrm{Prop} | \mathrm{Prf}(M)$ and $M, N ::= x | \star | 0 | \mathrm{Suc}(M) | \mathrm{R}^{\mathrm{Nat}}_{\sigma}(M, N, O) | (M\ N) | (\lambda x{:}\sigma.M) | (M, N) | M.1 | M.2 | \forall x{:}\sigma.M$ The empty context is valid and if $\Gamma \vdash \sigma_{\mathrm{set}}$ then $\Gamma, x{:}\sigma$ **ctxt** for x fresh.Nat, 1, and Prop are valid types in every valid context. If $\Gamma, x{:}\sigma \vdash \tau$ **type** then $\Gamma \vdash \Pi x{:}\sigma.\tau$ **type** and $\Gamma \vdash \Sigma x{:}\sigma.\tau$ **type**. If $\Gamma \vdash M : \mathrm{Prop}$ then $\Gamma \vdash \mathrm{Prf}(M)$ **type**. The term forming rules are as follows.

$$\frac{\Gamma \vdash \sigma = \tau\ \mathbf{type} \qquad \Gamma \vdash M : \sigma}{\Gamma \vdash M : \tau} \qquad \frac{\Gamma, x{:}\sigma, \Delta\ \mathbf{ctxt}}{\Gamma, x{:}\sigma, \Delta \vdash x{:}\sigma}$$

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau[M]}{\Gamma \vdash (M, N) : \Sigma x{:}\sigma.\tau[x]} \qquad \frac{\Gamma \vdash M : \Sigma x{:}\sigma.\tau[x]}{\Gamma \vdash M.1 : \sigma} \qquad \frac{\Gamma \vdash M : \Sigma x{:}\sigma.\tau[x]}{\Gamma \vdash M.2 : \tau[M.1]}$$

$$\frac{\Gamma, x{:}\sigma \vdash M : \tau[x]}{\Gamma \vdash \lambda x{:}\sigma.M : \Pi x{:}\sigma.\tau[x]} \qquad \frac{\Gamma \vdash M : \Pi x{:}\sigma.\tau[x] \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\ N) : \tau[N]}$$

$$\frac{\Gamma\ \mathbf{ctxt}}{\Gamma \vdash 0 : \mathrm{Nat}} \qquad \frac{\Gamma \vdash M : \mathrm{Nat}}{\Gamma \vdash \mathrm{Suc}(M) : \mathrm{Nat}} \qquad \frac{\Gamma, x{:}\mathrm{Nat} \vdash \sigma[x]\ \mathbf{type} \quad \Gamma \vdash M_z : \sigma[0] \quad \Gamma \vdash N{:}\mathrm{Nat} \quad \Gamma, x{:}\mathrm{Nat}, p{:}\sigma[x] \vdash M_s : \sigma[\mathrm{Suc}(x)]}{\Gamma \vdash \mathrm{R}^{\mathrm{Nat}}_{\sigma}(M_z, M_s, N) : \sigma[N]}$$

$$\frac{\Gamma\ \mathbf{ctxt}}{\Gamma \vdash \star : 1} \qquad \frac{\Gamma, x{:}\sigma \vdash P : \mathrm{Prop}}{\Gamma \vdash \forall x{:}\sigma.P : \mathrm{Prop}}$$

A.1 Equality Rules

Equality is reflexive, symmetric, and transitive and for each context, term, or type forming rule there is a corresponding equality rule. For example

$$\frac{\Gamma \vdash \sigma = \sigma'\ \mathbf{type} \qquad \Gamma, x : \sigma \vdash \tau = \tau'\ \mathbf{type}}{\Gamma \vdash \Pi x : \sigma.\tau = \Pi x : \sigma'.\tau'\ \mathbf{type}}$$

In addition we have the reflection rule for propositions $\Gamma \vdash \mathrm{Prf}(\forall x{:}\sigma.P) = \Pi x{:}\sigma.\mathrm{Prf}(P)$ **type** and the following "computational" equations which are understood to hold under the premise that both left and right hand side have the appropriate type: $\Gamma \vdash \mathrm{App}_{\sigma,\tau}(\lambda x : \sigma.M, N) = M[N] : \tau[N]$, $\Gamma \vdash (M, N).1 = M : \sigma$, $\Gamma \vdash (M, N).2 = N : \tau[M]$, $\Gamma \vdash M = \star : 1$ (rule for the extensional unit type), $\Gamma \vdash \mathrm{R}^{\mathrm{Nat}}_{\sigma[x:\mathrm{Nat}]}(M_z\ , M_s[x{:}\mathrm{Nat}, p{:}\sigma[x]]\ , 0) = M_z : \sigma[0]$, and $\Gamma \vdash \mathrm{R}^{\mathrm{Nat}}_{\sigma[x:\mathrm{Nat}]}(M_z\ , M_s[x{:}\mathrm{Nat}, p{:}\sigma[x]]\ , \mathrm{Suc}(N)) = M_s[N, \mathrm{R}_{\sigma}(M_z, M_s, N)] : \sigma[\mathrm{Suc}(N)]$.

This article was processed using the LaTeX macro package with LLNCS style