CATEGORICAL MODELS OF TYPE THEORY

AUKE BOOIJ AND DANIIL FRUMIN

Contents

1. Introduction

In type theory, theorems are interpreted by types, and proofs of theorems are interpreted as elements of those theorems. Hence, in this logical view, we prove theorems of type theory by finding elements of certain types. Now dependent type theories can be very expressive, and in particular interpret Peano Arithmetic (PA). Hence, we can apply Gödel's first incompleteness theorem to deduce that there must be a "logical sentence" (namely a type) G such that neither G nor $\neg G = G \to \mathbf{0}$ is inhabited.

More generally, a book on the *theory* of dependent type theory, such as the Homotopy Type Theory book [13], teaches nothing about the *proof strength* of dependent type theories. For example, it was only the introduction of *univalence* axioms that introduced provable existence of nontrivial elements of the identity type.

Date: June 2015.

Hence, we might wonder if, just as in classical logic, there exists a framework for proving that certain types are uninhabited. This is the topic of this essay.

We will see that there *is* such a notion (in fact, several!), and we will apply them to prove the independence of notions.

For illustrative purposes, we will prove that in dependent type theory with natural numbers but *without* universes, we cannot prove the third Peano axiom, that states that $0 \neq 1$ (see subsection 2.5). We will also look at the groupoid model in subsection 4.1, that showed that the existence of nontrivial identity terms is independent of plain dependent type theory (i.e. with universes).

1.1. **Soundness and completeness of models.** In classical logic (e.g. *first-order logic*), the framework for proving independence of logical claims is given by *models*: we find two models of a fixed theory T, one in which a sentence ϕ is true, and one in which φ is false (i.e. $\neg\phi$ is true).

Now suppose that we have those. Then recall that the soundness theorem for first-order logic states that if a logical sentence is provable in a theory T, then in fact it is true in every model that makes all sentences of T true.

Hence, if φ were provable, then the fact that we have a model in which it is false contradicts the provability of φ. The fact that we also have a model in which it is true now gives us that φ must be independent of T.

Now a result for models of first-order logic that is typically seen as more interesting (but, as per the above, not more important *per se*) is the completeness theorem. This gives the converse to soundness: if a sentence φ is true in all models of a theory T, then φ itself is *provable* from T.

In the models of type theory that we will study, the completeness result will be a very easy result: it follows from a *syntactic* model of type theory that models exactly those things that are provable. In fact, soundness is now a somewhat nontrivial result. We will discuss these results later in section 3.

2. Categories with families

To define our notion of model, we let ourselves be inspired by the syntactic side of things. This link is made more precise in subsection 2.6.

2.1. **Category of contexts.** One early "naive" notion of models of type theory is given by a category $\mathcal{C}$ of contexts and context morphisms. In the syntactical ("theoretical") side of this, we have that substitutions represent context morphisms.

Given such a category, we will want to have a mechanism that, given a context Γ and a type σ in Γ, gives an extended context $\Gamma.\sigma$. Syntactically, this would be represented by extending the list of types by $x : \sigma$, yielding $\Gamma, x : \sigma$. Also, syntactically, context extensions of Γ correspond bijectively with "basic projection morphisms" $\mathsf{p}(\sigma) : \Gamma.\sigma \to \Gamma$ to Γ. So define a *naive model of type theory* to be a category $\mathcal{C}$ together with a set of morphisms B that are thought of as basic dependent projections. We write morphisms $f : \Theta \to \Gamma$ in B as $\mathsf{p}(\sigma) : \Gamma.\sigma \to \Gamma$ where Γ is the codomain of f and σ is an arbitrary symbol, thought of as the type specified by f.

In the syntactic case, the terms of type σ in a given context Γ correspond bijectively to sections of $\mathsf{p}(\sigma)$. By analogy with this, we can define semantic terms in a fixed context Γ to be the set of sections of the projection morphism $\mathsf{p}(\sigma) : \Gamma.\sigma \to \Gamma$.

We can hope to interpret type theory in naive models of type theory in this way, but unfortunately we get *degeneracy issues*: we require more properties on the model to be able to prove a soundness theorem (we will see later exactly what soundness means for type theory).

More specifically, suppose that we have a context morphism $f : \Gamma \to \Delta$, and that we have a type σ in the context Δ (represented by a basic dependent projection $\mathsf{p}(\sigma)$). Then we would like to extend f between the contexts that are extended by σ. In other words, we would like to find f' in the pullback square

$$\begin{array}{ccc} \Gamma' & \xrightarrow{f'} & \Delta.\sigma \\ \downarrow{\scriptstyle p} & & \downarrow{\scriptstyle \mathsf{p}(\sigma)} \\ \Gamma & \xrightarrow[f]{} & \Delta \end{array}$$

such that p becomes a basic dependent projection. However, purely from categorical notions, we cannot uniquely find Γ', even if $\mathcal{C}$ has pullbacks, since limits in categories are only given up to isomorphism.

2.2. **Categories with families.** We solve this issue by working with explicit associations of sets of *types* to contexts, and sets of *terms* to types in contexts. This allows us to specify substitution principles for types and terms explicitly, so that substituted types and terms are unique.

More precisely, a *category with families* (CwF) [4, 6] is specified by the following data:

- $\mathcal{C}$**:** is the category of contexts and context morphisms with a fixed terminal object $\top$
- Ty **and** Tm**:** give the types in a context and the terms in a type (e.g. $\mathbf{N} \in \mathrm{Ty}(\Gamma)$ and $\mathsf{suc}(0) \in \mathrm{Tm}(\Gamma, \mathbf{N})$)
- $-\{-\}$**:** gives substitutions: for $f : \Gamma \to \Delta$ in $\mathcal{C}$ we have $-\{f\} : \mathrm{Ty}(\Delta) \to \mathrm{Ty}(\Gamma)$ and $-\{f\} : \mathrm{Tm}(\Delta, A) \to \mathrm{Tm}(\Gamma, A\{f\})$
- $\langle\rangle\_$**:** gives the unique morphism $\langle\rangle_\Gamma : \Gamma \to \top$ to the empty context $\top \in \mathcal{C}$
- $-.-$**:** is context extension, giving for each context $\Gamma \in \mathrm{ob}\,\mathcal{C}$ and each $\sigma \in \mathrm{Ty}(\Gamma)$ a new context $\Gamma.\sigma$
- p**:** projects away the last type, so $\mathsf{p}(\Gamma.\sigma) : \Gamma.\sigma \to \Gamma$ in $\mathcal{C}$ (where the Γ parameter can be either implicit or explicit, so $\mathsf{p}(\sigma) = \mathsf{p}(\Gamma.\sigma)$)
- $\mathsf{v}\_$**:** is the other projection: for each $\sigma \in \mathrm{Ty}(\Gamma)$ it gives a term v_σ with $\mathsf{v}_\sigma \in \mathrm{Tm}(\Gamma.\sigma, \sigma\{\mathsf{p}(\sigma)\})$
- $\langle -, - \rangle\_$**:** is a context morphism extension: if $f : \Gamma \to \Delta$ and $M \in \mathrm{Tm}(\Gamma, \sigma\{f\})$ then $\langle f, M \rangle_\sigma : \Gamma \to \Delta.\sigma$

that satisfies certain *sanity conditions*. Morally, the only reason we require these conditions is that we want to arrive at a notion of models of type theory that is, on the one hand, sufficiently rich (i.e. allows nontrivial models that we can use to show e.g. independence), but, on the other hand, is sound for type theory (as we require a soundness theorem for such independence results). However, we can certainly verify that the laws below are indeed satisfied in the syntactic case, and this is the motivation for this specific choice of laws.

Specifically, if

- $\Gamma, \Delta, \Theta \in \mathrm{ob}\,\mathcal{C}$
- $f : \Gamma \to \Delta$ and $g : \Delta \to \Theta$

- $\sigma \in \mathrm{Ty}(\Theta)$
- $M \in \mathrm{Tm}(\Theta, \sigma)$
- $N \in \mathrm{Tm}(\Delta, \sigma\{g\})$

then we require that

$$\begin{aligned}
\sigma\{\mathrm{id}_\Theta\} &= \sigma && \in \mathrm{Ty}(\Theta) \\
\sigma\{g \circ f\} &= \sigma\{g\}\{f\} && \in \mathrm{Ty}(\Gamma) \\
M\{\mathrm{id}_\Theta\} &= M && \in \mathrm{Tm}(\Theta, \sigma) \\
M\{g \circ f\} &= M\{g\}\{f\} && \in \mathrm{Tm}(\Gamma, \sigma\{g \circ f\}) \\
\mathsf{p}(\Theta.\sigma) \circ \langle g, N \rangle_\sigma &= g && : \Delta \to \Theta \\
\mathsf{v}_\sigma\{\langle g, N \rangle_\sigma\} &= N && \in \mathrm{Tm}(\Delta, \sigma\{g\}) \\
\langle g, N \rangle_\sigma \circ f &= \langle g \circ f, N\{f\} \rangle_\sigma && : \Gamma \to \Theta.\sigma \\
\langle \mathsf{p}(\Theta.\sigma), \mathsf{v}_\sigma \rangle_\sigma &= \mathrm{id}_{\Theta.\sigma} && : \Theta.\sigma \to \Theta.\sigma
\end{aligned}$$

From the given laws it can be seen that Ty is a presheaf over $\mathcal{C}$, with $\mathrm{Ty}(f : A \to B)$ defined as $-\{f\} : \mathrm{Ty}(B) \to \mathrm{Ty}(A)$.

We also refer to a given CwF by referring to its category of contexts $\mathcal{C}$, leaving the remaining structure implicit.

The above definition of Categories with Families tells us how to semantically interpret contexts and substitutions. However, to even be able to state a soundness theorem, we need to know what it means to interpret types and terms. We will discuss this in subsection 2.4, but first we will look at a few technicalities surrounding context morphisms and its interaction with terms: namely, how we can *weaken* context morphisms to act on larger contexts, and how context morphisms can represent terms.

2.3. **Terms and sections.** Given a term $M \in \mathrm{Tm}(\Gamma, \sigma)$, we can construct a *section* of $\mathsf{p}(\Gamma.\sigma)$, given by[1]

$$\overline{M} = \langle \mathrm{id}_\Gamma, M \rangle : \Gamma \to \Gamma.\sigma$$

Given the interplay of p and $\langle -, - \rangle$, it is easy to see that $\overline{M}$ is indeed a section. Conversely, given a section s of $\mathsf{p}(\Gamma.\sigma)$, we can construct a term

$$\begin{aligned}
\mathsf{v}_\sigma\{s\} \in \mathrm{Tm}(\Gamma.\sigma, \sigma\{\mathsf{p}(\sigma)\})\{s\} &= \mathrm{Tm}(\Gamma, \sigma\{\mathsf{p}(\sigma)\}\{s\}) \\
&= \mathrm{Tm}(\Gamma, \sigma\{\mathsf{p}(\sigma) \circ s\}) \\
&= \mathrm{Tm}(\Gamma, \sigma\{\mathrm{id}_\Gamma\}) \\
&= \mathrm{Tm}(\Gamma, \sigma)
\end{aligned}$$

Furthermore, the two aforementioned operations are inverses of each other:

$$\mathsf{v}_\sigma\{\overline{M}\} = M$$

[1]When unambiguous, we drop the last argument to $\langle -, - \rangle_-$, writing $\langle \mathrm{id}_\Gamma, M \rangle$ instead of $\langle \mathrm{id}_\Gamma, M \rangle_\sigma$

$$\begin{aligned}\overline{\mathsf{v}_\sigma\{s\}} = \langle \mathrm{id}_\Gamma, \mathsf{v}_\sigma\{s\}\rangle &= \langle \mathsf{p}(\sigma)\circ s, \mathsf{v}_\sigma\{s\}\rangle \\ &= \langle \mathsf{p}(\sigma), \mathsf{v}_\sigma\rangle \circ s \\ &= \mathrm{id}_{\Gamma.\sigma}\circ s = s\end{aligned}$$

It follows that we can identify terms and sections; sometimes, abusing the notation, we will speak of terms as if they were sections and vice versa.

2.4. **Interpreting types.** Establishing that a category $\mathcal{C}$ has a structure of a category with families is not enough to start interpreting type theory inside $\mathcal{C}$. After all, we need to know how to interpret specific type formers. The requirements for a CwF to support certain types are closely related to the syntactic rules for said types.

A crucial definition that is used throughout this section is the one of weakening:

Given a context morphism $f : \Gamma \to \Delta$ and a type $\sigma \in \mathrm{Ty}(\Delta)$, we can *weaken* f by σ, and obtain a morphism $q(f,\sigma) : \Gamma.\sigma\{f\} \to \Delta.\sigma$. The weakening is defined as

$$q(f,\sigma) = \langle f \circ \mathsf{p}(\sigma\{f\}), \mathsf{v}_{\sigma\{f\}}\rangle_\sigma$$

This morphism also makes the following square a pullback:

$$\begin{array}{ccc} \Gamma.\sigma\{f\} & \xrightarrow{q(f,\sigma)} & \Delta.\sigma \\ \downarrow{\scriptstyle \mathsf{p}(\sigma\{f\})} & & \downarrow{\scriptstyle \mathsf{p}(\sigma)} \\ \Gamma & \xrightarrow[f]{} & \Delta \end{array}$$

Thus we reobtain the notion of pullback as a substitution operation.

2.4.1. Π *types.* Consider the Π-types. The requirements for the CwF to support Π-types (just like for any times) can be divided into several groups: formation, introduction, and elimination.

A CwF supports Π-types if for any context Γ and for any two types $\sigma \in \mathrm{Ty}(\Gamma)$, and $\tau \in \mathrm{Ty}(\Gamma.\sigma)$

(1) There is a type $\Pi(\sigma,\tau) \in \mathrm{Ty}(\Gamma)$
(2) For any $M \in \mathrm{Tm}(\Gamma.\sigma,\tau)$, there is a term $(\lambda_{\sigma,\tau})(M) \in \mathrm{Tm}(\Gamma, \Pi(\sigma,\tau))$
(3) There is a morphism $App_{\sigma,\tau} : \Gamma.\sigma.\Pi(\sigma,\tau)\{\mathsf{p}(\sigma)\} \to \Gamma.\sigma.\tau$, such that

$$\mathsf{p}(\Gamma.\sigma.\tau)\circ App_{\sigma,\tau} = \mathsf{p}(\Gamma.\sigma.\Pi(\sigma,\tau))$$

(i.e. App preserves the initial part of the context), and

$$App_{\sigma,\tau}\circ \overline{(\lambda_{\sigma,\tau}(M))\{\mathsf{p}(\Gamma.\sigma)\}} = \overline{M}$$

for any $M \in \mathrm{Tm}(\Gamma.\sigma,\tau)$ (the computation rule)
(4) All of the aforementioned constructs are stable under substitutions

The last point requires a bit of elaboration. We will show how to derive the "stability" laws using the Π type former as an example.

Suppose there is a morphism $f : \Delta \to \Gamma$, and a type $\Pi(\sigma, \tau) \in \mathrm{Ty}(\Gamma)$. So we have the diagram:

$$\begin{array}{ccc} & & \Gamma.\Pi(\sigma,\tau) \\ & & \downarrow{\scriptstyle \mathsf{p}(\Pi(\sigma,\tau))} \\ \Delta & \xrightarrow{f} & \Gamma \end{array}$$

It follows that $\sigma \in \mathrm{Ty}(\Gamma)$ and $\tau \in \mathrm{Ty}(\Gamma.\sigma)$. We have two ways of pulling back $\Pi(\sigma, \tau)$ along f. First of all, we can just pull the whole type back:

$$\begin{array}{ccc} \Delta.\Pi(\sigma,\tau)\{f\} & \xrightarrow{q(f,\Pi(\sigma,\tau))} & \Gamma.\Pi(\sigma,\tau) \\ \downarrow{\scriptstyle \mathsf{p}(\Pi(\sigma,\tau)\{f\})} & & \downarrow{\scriptstyle \mathsf{p}(\Pi(\sigma,\tau))} \\ \Delta & \xrightarrow{f} & \Gamma \end{array}$$

Secondly, we can pull back the constituents of the Π type — σ and τ — and then put them back together using the Π type constructor:

$$\begin{array}{ccc} \Delta.\sigma\{f\}.\tau\{q(f,\sigma)\} & \xrightarrow{q(q(f,\sigma),\tau)} & \Gamma.\sigma.\tau \\ \downarrow{\scriptstyle \mathsf{p}(\tau\{q(f,\sigma)\})} & & \downarrow{\scriptstyle \mathsf{p}(\tau)} \\ \Delta.\sigma\{f\} & \xrightarrow{q(f,\sigma)} & \Gamma.\sigma \\ \downarrow{\scriptstyle \mathsf{p}(\sigma\{f\})} & & \downarrow{\scriptstyle \mathsf{p}(\sigma)} \\ \Delta & \xrightarrow{f} & \Gamma \end{array}$$

yielding the types

$$\begin{array}{rr} & \sigma\{f\} \in \mathrm{Ty}(\Delta) \\ \text{hence} & \tau\{q(f,\sigma)\} \in \mathrm{Ty}(\Delta.\sigma\{f\}) \\ \text{finally giving} & \Pi(\sigma\{f\}, \tau\{q(f,\sigma)\}) \in \mathrm{Ty}(\Delta) \end{array}$$

The "stability" law states that those two ways of obtaining a Π type over Δ are equivalent:

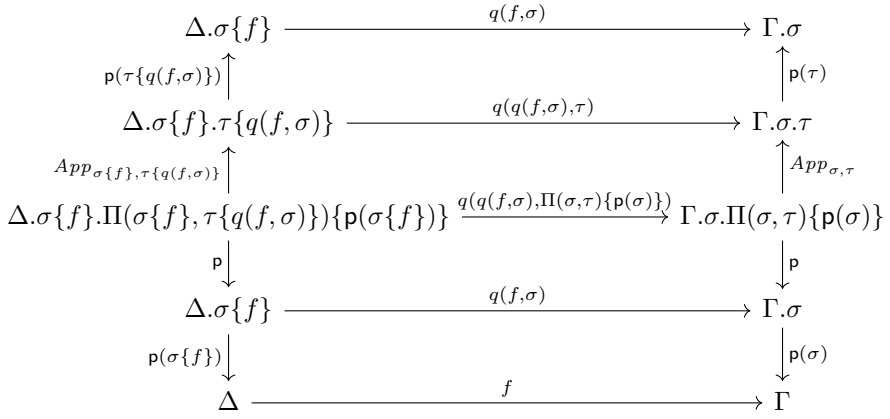
$$\Pi(\sigma,\tau)\{f\} = \Pi(\sigma\{f\}, \tau\{q(f,\sigma)\})$$

Further down the text we will not work out all of the details for other stability laws, but we will merely state them. An interested reader can reproduce the diagrams that lead to the specific laws.

The stability under substitutions are also required for λ and App:

(1) $(\lambda_{\sigma,\tau})(M)\{f\} = (\lambda_{\sigma\{f\},\tau\{q(f,\sigma)\}})(M\{q(f,\sigma)\})$
(2) $App_{\sigma,\tau} \circ q(q(f,\sigma), \Pi(\sigma,\tau)\{\mathsf{p}(\sigma)\}) = q(q(f,\sigma),\tau) \circ App_{\sigma\{f\},\tau\{q(f,\sigma)\}}$

The App substitution stability laws expresses the commutativity of the penultimate upper square in the following diagram:



The diagram makes sense because the equation

$$\Delta.\sigma\{f\}.\Pi(\sigma\{f\},\tau\{q(f,\sigma)\})\{\mathsf{p}(\sigma\{f\})\} = \Delta.\sigma\{f\}.\Pi(\sigma,\tau)\{\mathsf{p}(\sigma)\circ q(f,a)\}$$

holds by the stability of Π under substitutions.

2.4.2. Σ *types.* A CwF supports Σ types if for any context Γ and for any two types $\sigma \in \mathrm{Ty}(\Gamma)$, and $\tau \in \mathrm{Ty}(\Gamma.\sigma)$

(1) There is a type $\Sigma(\sigma,\tau) \in \mathrm{Ty}(\Gamma)$
(2) There is a morphism $Pair_{\sigma,\tau} : \Gamma.\sigma.\tau \to \Gamma.\Sigma(\sigma,\tau)$, such that

$$\mathsf{p}(\Sigma(\sigma,\tau)) \circ Pair_{\sigma,\tau} = \mathsf{p}(\sigma)\circ\mathsf{p}(\tau)$$

(3) For every $\rho \in \mathrm{Ty}(\Gamma.\Sigma(\sigma,\tau))$ and $H \in \mathrm{Tm}(\Gamma.\sigma.\tau, \rho\{Pair_{\sigma,\tau}\})$ there is a term $R^{\Sigma}(H) \in \mathrm{Tm}(\Gamma.\Sigma(\sigma,\tau),\rho)$ such that

$$R^{\Sigma}(H)\{Pair_{\sigma,\tau}\} = H$$

and for any morphism $f : \Delta \to \Gamma$

(1) $\Sigma(\sigma,\tau)\{f\} = \Sigma(\sigma\{f\}, \tau q(f,\sigma))$
(2) $q(f,\Sigma(\sigma,\tau)) \circ Pair_{\sigma\{f\},\tau q(f,\sigma)} = Pair_{\sigma,\tau}\circ q(q(f,\sigma),\tau)$

2.4.3. *Empty type.* A CwF supports Σ types if for any context Γ,

(1) There is a type $\mathbf{0} \in \mathrm{Ty}(\Gamma)$, such that $\mathbf{0}\{f\} = \mathbf{0}$
(2) Since the theory has no constructors for $\mathbf{0}$, we have no requirements on the existence of terms of type $\mathbf{0}$.
(3) For any $\rho \in \mathrm{Ty}(\Gamma.\mathbf{0})$, there is a term $R^{\mathbf{0}}_{\rho} \in \mathrm{Tm}(\Gamma.\mathbf{0},\rho)$.

2.4.4. *Unit type.* A CwF supports the unit type if for any context Γ,

(1) There is a type $\mathbf{1} \in \mathrm{Ty}(\Gamma)$, such that $\mathbf{1}\{f\} = \mathbf{1}$
(2) There is a morphism $* : \Gamma \to \Gamma.\mathbf{1}$, such that $\mathsf{p}(\mathbf{1}) \circ * = \mathrm{id}_{\Gamma}$
(3) For any $\rho \in \mathrm{Ty}(\Gamma.\mathbf{1})$, there is a term $R^{\mathbf{1}}_{\rho} : \mathrm{Tm}(\Gamma, \rho\{*\}) \to \mathrm{Tm}(\Gamma.\mathbf{1},\rho)$, such that

$$R^{\mathbf{1}}_{\rho}(H)\{*\} = H$$

for $H \in \mathrm{Tm}(\Gamma, \rho\{*\})$.

2.4.5. *Identity types.* A CwF supports identity types if for any context Γ, and for a type $A \in \mathrm{Ty}(\Gamma)$,

- There is a type $\mathsf{Id}_A \in \mathrm{Ty}(\Gamma.A.A\{\mathsf{p}(A)\})$, that is stable under substitutions, i.e. if $f : \Delta \to \Gamma$ is a morphism, then
$$\mathsf{Id}_A\{q(q(f, A), A\{\mathsf{p}(A)\})\} = \mathsf{Id}_{A\{f\}}$$
- There is a morphism $\mathit{Refl}_A : \Gamma.A \to \Gamma.A.A\{\mathsf{p}(A)\}.\mathsf{Id}_A$, such that
$$\mathsf{p}(\mathsf{Id}_A) \circ \mathit{Refl}_A = \overline{\mathsf{v}_\sigma}$$
and Refl respects substitution
- For every $\tau \in \mathrm{Ty}(\Gamma.A.A\{\mathsf{p}(A)\}.\mathsf{Id}_A)$ and $H \in \mathrm{Tm}(\Gamma.A, \tau\{\mathit{Refl}_A\})$, there is a term $R^{\mathsf{Id}}(H) \in \mathrm{Tm}(\Gamma.A.A\{\mathsf{p}(A)\}, \tau)$ such that
  - $R^{\mathsf{Id}}(H)\{\mathit{Refl}\} = H$
  - $q(q(q(f, A), A\{\mathsf{p}(A)\}), \mathsf{Id}_A) \circ \mathit{Refl}_{A\{f\}} = \mathit{Refl}_A \circ q(f, A)$

2.4.6. *Natural numbers.* A CwF supports natural numbers if for any context Γ,

(1) There is a type $\mathbf{N} \in \mathrm{Ty}(\Gamma)$, that is stable under substitutions (i.e. $\mathbf{N}\{f\} = \mathbf{N}$);
(2) There are morphisms $O : \Gamma \to \Gamma.\mathbf{N}$ and $S : \Gamma.\mathbf{N} \to \Gamma.\mathbf{N}$, such that
$$\begin{aligned} \mathsf{p}(\Gamma.\mathbf{N}) \circ O &= \mathrm{id}_\Gamma \\ \mathsf{p}(\Gamma.\mathbf{N}) \circ S &= \mathsf{p}(\Gamma.\mathbf{N}) \end{aligned}$$
(3) For each $\tau \in \mathrm{Ty}(\Gamma.\mathbf{N})$, there is a term
$$R^{\mathbf{N}}_\tau : \mathrm{Tm}(\Gamma, \tau\{O\}) \to \mathrm{Tm}(\Gamma.\mathbf{N}.\tau, \tau\{S \circ \mathsf{p}(\Gamma.\mathbf{N}.\tau)\}) \to \mathrm{Tm}(\Gamma.\mathbf{N}, \tau)$$
such that
$$\begin{cases} R^{\mathbf{N}}_\tau(P, Q)\{O\} = P \\ R^{\mathbf{N}}_\tau(P, Q)\{S\} = Q\{\overline{R^{\mathbf{N}}_\tau(P, Q)}\} \end{cases}$$
For $P \in \mathrm{Tm}(\Gamma, \tau\{O\})$, $Q \in \mathrm{Tm}(\Gamma.\mathbf{N}.\tau, \tau\{S \circ \mathsf{p}(\Gamma.\mathbf{N}.\tau)\})$.

2.5. **Example 1: the truth-valued model.** In the presense of universes, we can show $0 \neq 1$ to be true in the following way.

(1) Define a type family $F : \mathbf{N} \to \mathcal{U}$ by induction on $\mathbf{N}$ such that $F(0) \equiv \mathbf{1}$ and $F(\mathsf{suc}(x)) \equiv \mathbf{0}$.
(2) Assuming a path $p : 0 = 1$ we can transport $* : \mathbf{1} \equiv F(0)$ along p to get $p_*(*) : F(1) \equiv \mathbf{0}$.

Hence, in a dependent type theory with natural numbers and universes, it is provable that $0 \neq 1$. However, in a type theory without universes, this result is not provable, and hence the provability of $0 \neq 1$ is independent of such type theory. We will show this by defining the *truth-valued model*.

2.5.1. *Category B_2.* Consider a poset $\{\mathbb{ff}, \mathbb{tt}\}$ with $\mathbb{ff} \leq \mathbb{tt}$ as a category. This category, containing two objects and three morphisms will be denoted as B_2. Curiously, this category exhibits a CwF structure.

- $\mathbb{tt}$ is a terminal object;
- For each $\Gamma \in B_2$, $\mathrm{Ty}(\Gamma) = \{\mathbb{ff}, \mathbb{tt}\}$;

Type	Interpretation
$\Pi(A,B)$	$A \to B$ (Heyting implication)
$\Sigma(A,B)$	$A \wedge B$
Id_A	$\mathbb{tt}$
$\mathbf{N}$	$\mathbb{tt}$
$\mathbf{0}$	$\mathbb{ff}$

TABLE 1. Interpretation of type formers in the truth-valued model

- For each $\Gamma \in B_2$, $A \in \mathrm{Ty}(\Gamma)$, $\mathrm{Tm}(\Gamma, A) = \begin{cases} \{*\} & \text{if } \Gamma \leq A \\ \emptyset & \text{otherwise} \end{cases}$
- For each $\Gamma \in B_2$, $A \in \mathrm{Ty}(\Gamma)$, $\Gamma.A = \Gamma \wedge A$ (conjunction/meet)

2.5.2. *Interpreting types.* The interpretation of type formers is given in Table 1. It is straightforward to check that B_2 satisfies all the required laws.

Reader may notice, that the CwF and type formers strucutre on B_2 is defined in a way that is readily generalizable: in fact, the same definitions would work for any Heyting algebra, not just for two-element algebra. Thus, the models of Martin-Löf type theory are at least as rich as models of first-order intuitionistic logic.

2.5.3. *Application: Independence of the third Peano axiom.* The third Peano axiom states that $\neg(0 = 1)$. In the language of type theory it corresponds to the judgment

$$\vdash \mathsf{Id}_{\mathbf{N}}(0, \mathsf{suc}(0)) \to \mathbf{0} \text{ true}$$

However, the type $\mathsf{Id}_{\mathbf{N}}(0, \mathsf{suc}(0)) \to \mathbf{0}$ is not inhabited in B_2. One way to see that would be two consider the elements of $\mathbf{N}$ under the empty context: $\mathrm{Tm}(\mathbb{tt}, \mathbf{N})$. There is only one such element, namely $*$. In general, judgments of the form $p : \mathsf{Id}_A(a,b) \vdash t(p) : \mathbf{0}$ are not valid in B_2, because the proofs $t(p)$ of $\mathbf{0}$ in such judgments correspond to the elements of the set $\mathrm{Tm}(\mathsf{Id}_A\{\langle \bar{a}, b\rangle\}, \mathbf{0})$, which is empty in all non-trivial Heyting algebras, since $\mathbb{tt} \not\leq \mathbb{ff}$.

From this, and completeness of MLTT w.r.t. CwFs, it follows that the third Peano axiom is not dervable in MLTT without universes.

2.6. **Example 2: the term model.** A model of particular importance is the *term model*. It can be seen as a completely "syntactic" model of type theory, whose contexts, types and terms are exactly those syntactic objects we can prove to be valid contexts, types and terms.

More precisely, the category of contexts is given by comma-separated lists Γ of typed variables with the property that $\vdash \Gamma\ \mathsf{ctxt}$, where we identify such contexts Γ and Δ iff $\vdash \Gamma = \Delta\ \mathsf{ctxt}$. Morphisms between contexts Γ and Δ are syntactic morphisms from Γ to Δ: so they are typed lists of expressions f such that $\Gamma \vdash f \implies \Delta$. We identify $f, g : \Gamma \to \Delta$ iff $\Gamma \vdash f = g \implies \Delta$.

Given such a context Γ, we define $\mathrm{Ty}(\Gamma)$ to be those expressions σ such that $\Gamma \vdash \sigma\ \mathsf{type}$. Again, we quotient out definitional equality in the sense that we identify $\sigma, \tau \in \mathrm{Ty}(\Gamma)$ iff $\Gamma \vdash \sigma = \tau\ \mathsf{type}$. Given a context Γ and a type σ, the set of terms $\mathrm{Tm}(\Gamma, \sigma)$ is the set of equivalence classe (given by definitional equality) of syntactic terms M of type σ in context Γ.

Context extension is given syntactically: if $\vdash \Gamma\ \mathsf{ctxt}$ and $\Gamma \vdash \sigma\ \mathsf{type}$, then Γ is a list of typed variables $(x_1 : X_1, \ldots, x_n : X_n)$, and we define $\Gamma.\sigma$ to be the context

$(x_1 : X_1, \dots, x_n : X_n, a : \sigma)$, where a is a fresh variable (i.e. not occurring elsewhere in Γ).

All other structure follows similarly.

We use this model to prove completeness of our notion of models of type theory with respect to the theory. This is described further in subsection 3.2.

2.7. **Example 3: Set-theoretic model.** In the set-theoretic model, the underlying category $\mathcal{C}$ is the category of sets **Set**. For each set Γ, elements of $\mathrm{Ty}(\Gamma)$ are families of sets: $\{\sigma_\gamma\}_{\gamma\in\Gamma}$. Elements of $\mathrm{Tm}(\Gamma, \{\sigma_\gamma\}_{\gamma\in\Gamma})$ are "dependent functions" M, namely, an element of the cartesian product $\Pi_{\gamma\in\Gamma}\sigma_\gamma$, that assigns to $\gamma \in \Gamma$ an element of σ_γ.

The substitution operates on types as follows: if $f : \Delta \to \Gamma$ is a substitution, then

$$\{\sigma_\delta\}_{\delta\in\Delta}\{f\} := \{\sigma_{f(\delta)}\}_{\delta\in\Delta}$$

If $M \in \mathrm{Tm}(\Gamma, \sigma)$, then $M\{f\}(\delta) := M(f(\delta))$.

If $\sigma \in \mathrm{Ty}(\Gamma)$, then the context comprehension is defined as

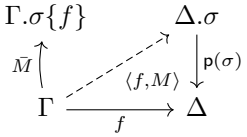
$$\Gamma.\sigma = \{(\gamma, x) \mid \gamma \in \Gamma, x \in \sigma_\gamma\}$$

In other words, as the disjoint union $\Gamma.\delta = \sqcup_\gamma \sigma_\gamma$. The projections p and v are defined as the first and the second projections, respectively:

$$\mathsf{p}(\Gamma.\sigma)(\gamma, x) = \gamma$$
$$\mathsf{v}_{\Gamma.\sigma}(\gamma, x) = x$$

(This "typechecks" because $\mathsf{v}_{\Gamma.\sigma}(\gamma, x) \in \sigma_{(\gamma,x)}\{\mathsf{p}(\sigma)\}$ and $\sigma_{(\gamma,x)}\{\mathsf{p}(\sigma)\} = \sigma_\gamma$.)

If $f : \Gamma \to \Delta$ is a morphism, and M is a term in $\mathrm{Tm}(\Gamma, \sigma\{f\})$, the extension of f is defined as $\langle f, M\rangle(\gamma) = (f(\gamma), M(\gamma))$.



Given a set Γ, a type $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$ the set theoretic model supports Σ-types with the following definitions:

- $\Sigma(A, B)_\gamma = \{(x, y) \mid x \in A_\gamma, y \in B_{(\gamma,x)}\}$
- $Pair_{A,B}(\gamma, a, b) = (\gamma, (a, b))$
- Given $\tau \in \mathrm{Ty}(\Gamma.\Sigma(A, B))$ and $H \in \mathrm{Tm}(\Gamma.A.B, \tau\{Pair\})$, the term $R^\Sigma(H)$ is defined as $R^\Sigma(H)(\gamma, (a, b)) = H(\gamma, a, b)$

Given $A \in \mathrm{Ty}(\Gamma)$, the set-theoretic model supports identity types with

- $(\mathsf{Id}_A)_{(\gamma,x,y)} = \begin{cases} \{*\} & \text{if } x = y \\ \emptyset & \text{otherwise} \end{cases}$
- $Refl_A(\gamma, x) = (\gamma, x, x, *)$
- Given $\tau \in \mathrm{Ty}(\Gamma.A.A.\mathsf{Id}_A)$, and $H \in \mathrm{Tm}(\Gamma.A, \tau\{Refl\})$, the recursor is defined as $R(H)(\gamma, x, y, p) = H(\gamma, x)$. (Note: this definition works because p can only be $*$ — in that case x and y are metatheoretically equal)

2.8. **Example 4: CwF from a presheaf category.** Given a category $\mathcal{C}$, we can assign a CwF structure to $\hat{\mathcal{C}}$ (the category of presheaves over $\mathcal{C}$).

In $\hat{\mathcal{C}}$, the contexts are presheaves, i.e. functors $\mathcal{C}^{op} \to \mathbf{Set}$, and the substitutions are natural transformations. The empty context is given by a terminal presheaf $\top$: a constant singleton $\top(x) = \{*\}$.

For a presheaf $\Gamma \in \hat{\mathcal{C}}$ we define $\mathrm{Ty}(\Gamma)$ to be the collection of presheaves over $\int(\Gamma)$, i.e. presheaves over a category of elements for Γ. If $f : \Delta \to \Gamma$, and $A \in \mathrm{Ty}(\Gamma)$, then we put $A\{f\} = A \circ \int(f) \in \mathrm{Ty}(\Delta)$. From the fact that $\int(-)$ is functorial, it is clear that $\mathrm{Ty}(-)$ follows all the necessary laws.

Next we define terms. Given a context Γ and a type $A \in \mathrm{Ty}(\Gamma)$, a term $M \in \mathrm{Tm}(\Gamma, A)$ is given by

- For each $I \in \mathcal{C}$ and $a \in \Gamma(I)$ an object $M(a) \in A(I, a)$;
- For each $I, J \in \mathcal{C}$, $u : J \to I$, $a \in \Gamma(I)$,

$$M(\Gamma(u)(a)) = A(u)(M(a))$$

Given a context Γ and a type $A \in \mathrm{Ty}(\Gamma)$, we define $\Gamma.A$ as:

$$\begin{aligned} \Gamma.A(I) &= \{(a, x) \mid a \in \Gamma(I), x \in A(I, a)\} && \text{(on objects)} \\ \Gamma.A(u : J \to I)(a, x) &= (\Gamma(u)(a), A(u)(x)) && \text{(on morphisms)} \end{aligned}$$

The projections are defined as the set-theoretic projections:

$$\mathsf{p}_I(a, x) = a$$

$$\mathsf{v}(I)(a, x) = x$$

Given a morphism $f : \Delta \to \Gamma$ and a term $M \in \mathrm{Tm}(\Delta, A\{f\})$, we define the context morphism extension as:

$$\langle f, M \rangle_I(\delta) = (f_I(\delta), M(I)(\delta))$$

Finally, we define how to interpret type formers in the presheaf model.

Given a context Γ and $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}(\Gamma.A)$, we define $\Pi(A, B)(I, a)$ as a set of families w, where each w is

$$w := \{w_f \mid J \in \mathcal{C}, f : J \to I\}$$

for $a \in \Gamma(I)$, where w_f is a "dependent function"

$$w_f \in \Pi_{u \in A(J, \Gamma(f)(a))} B(J, \Gamma(f)(a), u)$$

with the "naturality" requirement:

$$B(g)(w_f(u)) = w_{f \circ g}(A(g)(u))$$

The definition of $\Pi(A, B)$ on morphisms is the following: given $f : (I', \Gamma(f)(a)) \to (I, a)$, we put

$$\Pi(A, B)(f)(w) := \{w_{f \circ g} \mid g : J \to I'\}$$

Given $M \in \mathrm{Tm}(\Gamma.A, B)$ we put $\lambda(M) \in \mathrm{Tm}(\Gamma, \Pi(A, B))$ as

$$\lambda(M)(I, a) = w$$

where w is a family

$$w_{f:J\to I}(u) := M(J, \Gamma(f)(a), u)$$

Finally, the application morphism App is defined as

$$App_I(a, u, w) := w_{\mathrm{id}_I}(u)$$

where $a \in \Gamma(I), u \in A(I, a), w \in \Pi(A, B)(I, a)$.
The dependent sum $\Sigma(A, B)$ is defined as

$$\Sigma(A, B)(I, a) := \{(x, y) \mid x \in A(I, a), y \in B(I, a, x)\}$$

The constructor, recursor, and other details are given in [8, Section 1.2.2].

3. Soundness and completeness

We would like to state a soundness and completeness result for Categories with Families as models of type theory. We do this by defining an *interpretation function*. It maps the sentences of the theory, namely "syntactic" contexts, types and terms, to semantic such objects, namely objects of the category of contexts, elements of the set of types in a category, and elements of the set of terms in a type.

Now, every CwF $\mathcal{C}$[2] defines an interpretation function. In particular, we have three homonymous partial maps $[\![.]\!]_{\mathcal{C}}$ (where we will drop the CwF $\mathcal{C}$, so writing $[\![.]\!]$ instead) on the language of type theory such that

- for comma-separated lists Γ of typed variables, $[\![\Gamma]\!]$ becomes a semantic context;
- for expressions σ of type formers, $[\![\Gamma; \sigma]\!]$ becomes a semantic type in the context $[\![\Gamma]\!]$;
- for expressions M of introduction and elimination principles, $[\![\Gamma; M]\!]$ becomes a semantic term of an appropriate type $[\![\Gamma; \sigma]\!]$ in context $[\![\Gamma]\!]$.

Soundness is now stated as follows: for any CwF $\mathcal{C}$:

- If $\vdash \Gamma$ ctxt, then ($[\![\Gamma]\!]$ is defined and) $[\![\Gamma]\!] \in \mathrm{ob}\,\mathcal{C}$.
- If $\Gamma \vdash \sigma$ type, then $[\![\Gamma; \sigma]\!] \in \mathrm{Ty}([\![\Gamma]\!])$.
- If $\Gamma \vdash M : \sigma$, then $[\![\Gamma; M]\!] \in \mathrm{Tm}([\![\Gamma]\!], [\![\Gamma; \sigma]\!])$.

Additionally, soundness states that interpretation functions preserve equality in the following sense:

- If $\vdash \Gamma = \Delta$ ctxt, then $[\![\Gamma]\!] = [\![\Delta]\!]$.
- If $\Gamma \vdash \sigma = \tau$ type, then $[\![\Gamma; \sigma]\!] = [\![\Gamma, \tau]\!]$.
- If $\Gamma \vdash M = N : \sigma$, then $[\![\Gamma; M]\!] = [\![\Gamma; N]\!]$.

Conversely, we can state completeness:

- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma]\!]_{\mathcal{C}} \in \mathrm{ob}\,\mathcal{C}$, then $\vdash \Gamma$ ctxt.
- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma; \sigma]\!]_{\mathcal{C}} \in \mathrm{Ty}([\![\Gamma]\!])$, then $\Gamma \vdash \sigma$ type.
- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma; M]\!]_{\mathcal{C}} \in \mathrm{Ty}([\![\Gamma]\!], [\![\Gamma; \sigma]\!])$, then $\Gamma \vdash M : \sigma$.

And additionally:

- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma]\!] = [\![\Delta]\!]$, then $\vdash \Gamma = \Delta$ ctxt.
- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma; \sigma]\!] = [\![\Gamma; \tau]\!]$, then $\Gamma \vdash \sigma = \tau$ type.
- If for every CwF $\mathcal{C}$, we have that $[\![\Gamma; M]\!] = [\![\Gamma; N]\!]$, then $\Gamma \vdash M = N : \sigma$ for an appropriate type σ.

[2]Recall that we refer to Categories with Families by naming their underlying category of contexts, leaving the remaining structure implicit.

In the case of first-order logic, soundness is an easily verified result, which is generally considered to be straightforward. Somewhat curiously, soundness for our models of type theory is a nontrivial result, and in fact, completeness is immediate. For this reason, we start with discussing why completeness holds, and then describe the proof of soundness.

3.1. **Definition of interpretation functions.** To define such interpretation functions, we realize that its input is the language of type theory, which is defined inductively. Hence, a logical way to define them is by induction on the structure of the input: namely, on the structure of type-theoretical expressions.

The interpretation function actually consists of three (partial) maps, and hence we have to do at least three inductions. Additionally, by definition of the type theory, there are at least as many induction cases as there are type formers and introduction principles. So while the definition is not fundamentally complicated, there are many cases to consider.

For illustrative purposes, we list a few.[3]

$$\left.\begin{aligned} [\![\Diamond]\!] &:= \top \\ [\![\Gamma, x:\sigma]\!] &:= [\![\Gamma]\!].[\![\Gamma;\sigma]\!] \end{aligned}\right\} \text{ These define } [\![.]\!] \text{ for contexts.}$$

$$\left.\begin{aligned} [\![\Gamma;\Pi x:\sigma.\tau]\!] &:= \Pi([\![\Gamma;\sigma]\!],[\![\Gamma,x:\sigma;\tau]\!]) \\ &\vdots \end{aligned}\right\} \text{ Type formers}$$

$$\left.\begin{aligned} [\![\Gamma, x:\sigma; x]\!] &:= \mathsf{v}_{[\![\Gamma;\sigma]\!]} \\ [\![\Gamma, x:\sigma, \Delta, y:\tau; x]\!] &:= [\![\Gamma, x:\sigma, \Delta, x]\!]\{\mathsf{p}([\![\Gamma, x:\sigma, \Delta;\tau]\!])\} \\ [\![\Gamma; App_{\sigma,\tau}(M,N)]\!] &:= App_{[\![\Gamma;\sigma]\!],[\![\Gamma,x:\sigma;\tau]\!]}([\![\Gamma;M]\!],[\![\Gamma;N]\!]) \\ [\![\Gamma;\lambda x:\sigma.M^{\tau}]\!] &:= \lambda_{[\![\Gamma;\sigma]\!],[\![\Gamma,x:\sigma;\tau]\!]}([\![\Gamma,x:\sigma;M]\!]) \\ &\vdots \end{aligned}\right\} \text{ Terms}$$

3.2. **Completeness.** To prove completeness of our models of type theory, we need to, from knowledge of the models, somehow be able to exhibit proofs. In fact, we obtain these proofs trivially: for example, if some comma-separated list of typed variables is a valid context in any CwF, then in particular it is so in the term model of subsection 2.6. But we defined the contexts of the term model to be exactly those lists which are provable to be valid contexts (modulo definitional equality), so in particular this shows that the semantically valid context is a provably valid context, as required.

The cases for types and terms are completely analogous. Hence, we arrive at a completeness result for models of type theory.

3.3. **Soundness.** As noted in the introduction of this section, the proof for soundness is somewhat nontrivial. We can see why this must be the case: the definition of interpretation functions involves substitution principles, for example in the case of a term x, where x is not the last variable listed in the context. However, the

[3]Note that in the *App* case, the "function-application" notation is justified by the duality between terms and sections described in subsection 2.3.

conclusion given by the soundness theorem — namely that x is interpreted as a term of the type we expect — is free of any substitutions.

Hence, the proof of soundness must be able to adjust for these substitutions. This is typically done using a pair of *substitution lemmas*. Technical details of these can be found in Hofmann [6], and a more complete exposition is given in Streicher [12].

4. Other models

In sections 2.5, 2.6 and 2.7, we presented a few models. We now continue this discussion: we present models of particular significance to the development of Homotopy Type Theory.

4.1. **The groupoid model.** Of historical and conceptual importance is the model of dependent type theory in groupoids [7].

4.1.1. *Category of groupoids.* As the base category we consider the category **GPD** of *groupoids*. Given a groupoid/context Γ, the set of types over Γ is defined to be the set of functors from Γ to **GPD**

$$\mathrm{Ty}(\Gamma) = [\Gamma, \mathbf{GPD}]$$

A type $A \in \mathrm{Ty}(\Gamma)$ is also called a family of groupoids over Γ.

If $f : \Delta \to \Gamma$ is a groupoid morphism and $A \in \mathrm{Ty}(\Gamma)$, then $A\{f\}$ is defined simply as $A \circ f : \Delta \to \mathbf{GPD}$.

The set of terms $\mathrm{Tm}(\Gamma, A)$ is defined to be a set of *dependent objects* over A, i.e. objects M such that

- An object $M(\gamma) \in A(\gamma)$ for each $\gamma \in \Gamma$;
- A morphism $M(p) : A(p)(M(\gamma)) \to M(\gamma')$ for each $p : \gamma \to \gamma'$ in Γ, such that certain "functorial" laws hold:
  - $M(\mathrm{id}_\gamma) = \mathrm{id}_{M(\gamma)}$
  - $M(p' \circ p) = M(p') \circ (A(p')(M(p)))$

The substitution $M\{f\} \in \mathrm{Tm}(\Delta, A\{f\})$ is given by composing components of M with f.

Finally, we define the notions of context extension and context morphism extension.

If Γ is a groupoid and $A \in \mathrm{Ty}(\Gamma)$, then $\Gamma.A$ is defined to be a Grothendieck construction $\int A$ ("category of elements"); more explicitly $\Gamma.A$ is a groupoid of pairs (γ, a), such that $\gamma \in \Gamma$ and $a \in A(\gamma)$. A morphism between (γ, a) and (γ', a') is a pair

$$(\gamma, a) \xrightarrow{(p,q)} (\gamma', a')$$

such that $p : \gamma \to \gamma'$ in Γ and $q : A(p)(a) \to a'$ in $A(\gamma')$. The identity morphism is

$$\mathrm{id}_{(\gamma,a)} = (\mathrm{id}_\gamma, \mathrm{id}_a)$$

and the composition is defined as

$$(\gamma, a) \xrightarrow{(p,q)} (\gamma', a') \xrightarrow{(p',q')} (\gamma'', a'') \qquad \text{with} \quad (\gamma, a) \xrightarrow{(p' \circ p,\, q' \circ A(p')(q))} (\gamma'', a'')$$

The canonical projection $\mathsf{p}_{\Gamma.A} : \Gamma.A \to \Gamma$ is defined simply as $\mathsf{p}_A(\gamma, a) = \gamma$. The second projection $\mathsf{v}_A \in \mathrm{Tm}(\Gamma.A, A\{\mathsf{p}_A\})$ is defined as

$$\mathsf{v}_A(\gamma, a) = a$$
$$\mathsf{v}_A(p, q) = q$$

Given a morphism $f : \Delta \to \Gamma$ and $M \in \mathrm{Tm}(\Delta, A\{f\})$, we define a morphism $\langle f, M\rangle : \Delta \to \Gamma.A$ as

$$\langle f, M\rangle(\delta) = (f(\delta), M(\delta))$$
$$\langle f, M\rangle(p) = (f(p), M(p))$$

We know that $M(\delta) \in A\{f\}(\delta) = A(f(\delta))$, so the definition makes sense. Given this definition it is clear that $\mathsf{p}(A) \circ \langle f, M\rangle = f$, that $\mathsf{v}_A\{\langle f, M\rangle\} = M$, and that $\langle \mathsf{p}(A), \mathsf{v}_A\rangle = \mathrm{id}_{\Gamma.A}$; finally:

$$(\langle f, M\rangle \circ g)(\epsilon) = (f(g(\epsilon)), M(g(\epsilon))) = \langle f \circ g, M\{g\}\rangle(\epsilon)$$

4.1.2. *Interpreting types.* Arguably, one of the main contributions of the seminal paper [7] is the idea that types can be viewed as groupoids, and proofs of the identity types can be seen as isomorphisms in groupoids.

Consider an empty context $\Diamond$[4]. Then a type over $\Diamond$ is just a functor $A' := \Diamond \to \mathbf{GPD}$, which is to say, a groupoid $A := A'(*)$. We interpret the identity type $\mathsf{Id}_A(a, b)$ as a (discrete) groupoid $\mathrm{Hom}_A(a, b)$. The reflexivity term refl_a is interpreted as $\mathrm{id}_a \in \mathrm{Hom}_A(a, a)$. However, if we want Id_A to be a proper type, we have to extend it to a functor $A \times A \to \mathbf{GPD}$. The definition of Id_A on morphisms is the following: given $q_1 : a \to a'$, $q_2 : b \to b'$, we define $\mathsf{Id}_A(q_1, q_2) : \mathrm{Hom}_A(a, b) \to \mathrm{Hom}_A(a', b')$

$$\mathsf{Id}_A(q_1, q_2)(p) = q_2 \circ p \circ q_1^{-1}$$

$$\begin{array}{ccc} a & \xrightarrow{q_1} & a' \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle \mathsf{Id}_A(q_1,q_2)(p)} \\ b & \xrightarrow{q_2} & b' \end{array}$$

As for identity elimination, consider a context $[a : A, b : A, p : \mathsf{Id}_A(a, b)]$ (that is, $\Diamond.A.A\{\mathsf{p}_A\}.\mathsf{Id}_A$). The objects of such groupoid are triples (a, b, p). A morphism from (a, b, p) to (a', b', p') is a triple (q_1, q_2, s), where $q_1 : a \to a'$, $q_2 : b \to b'$, and $s : \mathsf{Id}_A(q_1, q_2)(p) \to p'$ — a morphism in $\mathsf{Id}_A(a', b')$. However, such a morphism s can exist only if $\mathsf{Id}_A(q_1, q_2)(p) = p'$, thus, we will omit the third component s from the triple.

In order to be able to interpret identity elimination, we have to derive a term $R(H)$ of a type $C : [a : A, b : A, p : \mathsf{Id}_A(a, b)] \to \mathbf{GPD}$ from a term H of a type $C\{\mathit{Refl}_A\} : [a : A] \to \mathbf{GPD}$. We put

$$R(H)(a, b, p) = C(\mathrm{id}_a, p, \mathrm{id}_p)(H(a))$$

[4]Note that this is the terminal groupoid, which has one element $*$ and one identity morphism.

Why does this work? Well, first of all, we note that $(\mathrm{id}_a, p, \mathrm{id}_p)$ is a morphism between (a, a, id_a) and (a, b, p), because $\mathsf{Id}_A(\mathrm{id}_a, p)(\mathrm{id}_a) = p$ (by the definition of $\mathsf{Id}_A(\mathrm{id}_a, p)$). Thus, $C(\mathrm{id}_a, p, \mathrm{id}_p) : C(a, a, \mathrm{id}_a) \to C(a, b, p)$. Finally, it is easy to check that $R(H)\{Refl_A\} = H$.

In order to define $R(H)$ on morphisms, let $p : a \to b$, let $p' : a' \to b'$, and let $(q_1, q_2) : (a, b, p) \to (a, b, p')$; then we need to define

$$R(H)(q_1, q_2) : C(q_1, q_2)(R(H)(a, b, p)) \to R(H)(a', b', p')$$

However, note that

$$C(q_1, q_2)(R(H)(a, b, p)) = C(q_1, q_2)(C(id_a, p)(H(a))) = C(q_1, q_2 \circ p)(H(a))$$

From the fact that $(q_1, q_2) : (a, b, p) \to (a, b, p')$ we get the commutativity of the following square:

$$\begin{array}{ccc} a & \xrightarrow{q_1} & a' \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p'} \\ b & \xrightarrow{q_2} & b' \end{array}$$

Thus

$$C(q_1, q_2 \circ p)(H(a)) = C(q_1, p' \circ q_1)(H(a)) = C(\mathrm{id}_{a'}, p')(C(q_1, q_1)(H(a)))$$

On the other hand, $R(H)(a', b', p') = C(\mathrm{id}_{a'}, p')(H(a'))$. Therefore, elaborating on type signature, we get

$$R(H)(q_1, q_2) : C(\mathrm{id}_{a'}, p')(C(q_1, q_1)(H(a))) \to C(id_{a'}, p')(H(a'))$$

It is thus sufficient to give a morphism $C(q_1, q_1)(H(a)) \to H(a')$, and lift it with $C(\mathrm{id}_{a'}, p')$. The morphism $H(q_1)$ is exactly such a morphism! In conclusion, we get the following definition:

$$R(H)(q_1, q_2) = C(\mathrm{id}_{a'}, p')(H(q_1))$$

It is straightforward to extend this definition to arbitrary contexts; technical details and further proofs are given in Hofmann and Streicher [7].

4.1.3. *Universe of groupoids.* We start by assuming a meta-theoretical universe $\mathcal{V}$. We call a groupoid X *small* if both objects and morphisms of X are in $\mathcal{V}$. By **Gpd** we denoted a category of small groupoids, restricting the morphisms to only isomorphisms of groupoids. Thus, **Gpd** is a groupoid itself: **Gpd** $\in$ **GPD**. **Gpd** will serve as a semantic universe object.

We want to have a type-theoretic universe U, such that $U \in \mathrm{Ty}(\Gamma)$ for any context Γ. We define U to be a constant functor sending each element $\gamma \in \Gamma$ to **Gpd** $\in$ **GPD**. Then a term of U in Γ (a "small type") would be a "dependent object" A, such that

(1) For all $\gamma \in \Gamma$, $A(\gamma)$ is an object of $U(\gamma)$;
(2) For all $f : \gamma \to \gamma'$, $A(f : \gamma \to \gamma')$ is a $U(\gamma)$-morphism from $U(f)(A(\gamma))$ and $A(\gamma')$;
(3) $A(f)$ has to satisfy the usual almost-functorial properties

Unfolding the definition of U we get

(1) For all $\gamma \in \Gamma$, $A(\gamma)$ is a small groupoid from **Gpd**;

(2) For all $f : \gamma \to \gamma'$, $A(f : \gamma \to \gamma') \in \mathrm{Hom}_{\mathbf{Gpd}}(A(\gamma), A(\gamma'))$, satisfying functorial laws.

As we can see, in the special case of U, a term $A \in \mathrm{Tm}(\Gamma, U)$ is just a functor $A : \Gamma \to \mathbf{Gpd}$. Finally, we want to turn terms A of the type U into types over Γ. For that consider a (faithful, but not full) inclusion functor $El : \mathbf{Gpd} \to \mathbf{GPD}$. Then $El \circ A : \Gamma \to \mathbf{GPD}$ is a type over Γ. Such types are called "small".

One might wonder if we get $U : U$ in that system? Well, if the meta-universe $\mathcal{V}$ was an actual universe, then $\mathbf{Gpd} \notin \mathbf{Gpd}$. Hence, U is not a functor from $\Gamma \to \mathbf{Gpd}$, and, therefore, the type-theoretic universe is not a term of itself.

Using the El functor, we can construct a *universe* structure in the sense of Kapulkin et al. [9]. We define U as $\mathbf{Gpd}$ and $\tilde{U}$ as the Grothendieck construction for El: namely as $G(\mathbf{Gpd}, El)$. This automatically gives us a fibration $p : G(\mathbf{Gpd}, El) \to \mathbf{Gpd}$ of groupoids. Then, for each groupoid Γ and a morphism $A : \Gamma \to \mathbf{Gpd}$, we have that $\mathsf{p}_A : \Gamma.A \to \Gamma$ is a pullback of p along A

$$\begin{array}{ccc} \Gamma.A & \xrightarrow{q} & G(\mathbf{Gpd}, El) \\ \downarrow{\scriptstyle \mathsf{p}_A} & & \downarrow{\scriptstyle p} \\ \Gamma & \xrightarrow{A} & \mathbf{Gpd} \end{array}$$

where $q(\gamma, a) = (A(\gamma), a)$.

In order to see that it is a pullback, consider an arbitrary X and morphisms $f : X \to G(\mathbf{Gpd}, El)$ and $h : X \to \Gamma$, such that $p \circ f = A \circ h$. That implies that for every $x \in X$, we have that $f(x)$ has the form $(A(h(x)), y)$ for $y \in A(h(x))$. Then define $k : X \to \Gamma.A$ as $k(x) = (h(x), y)$.

4.2. **Cubical sets.** The model of dependent type theory in cubical sets [2, 8] is of importance because it is a *constructive* model of a type theory with universe *univalence*. Constructing such a model is a first step in providing a computational explanation for univalence, and turning Homotopy Type Theory into a programming language.

4.2.1. *Geometrical presentation.* Cubical sets are defined using the *cube category*, denoted by $\square$. It consists of sets $I^0, I^1, I^2, \ldots$, where $I = \{0, 1\}$ is an "interval". Thus, I^0 can be seen as a point, I^1 can be seen as a line, and so on. The morphisms in $\square$ are generated by two classes of morphisms: face maps and degeneracy maps.

Face maps are maps $\delta^\epsilon_i(n) : I^n \to I^{n+1}$ of the form

$$\delta^\epsilon_i(n)(x_1, \ldots, x_n) = (x_1, \ldots, x_{i-1}, \epsilon, x_i, \ldots, x_n)$$

for $\epsilon = 0, 1$, and $1 \leq i \leq n+1$. Face maps can be seen as mappings of n-cubes to faces of $(n+1)$-cubes. The direction of the face is determine by i and the position is determined by ϵ.

The second class of maps are *degeneracy maps*: maps $e_i(n) : I^n \to I^{n-1}$ defined as

$$e_i(n)(x_1, \ldots, x_n) = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

for $1 \leq i \leq n$. Degeneracy maps can be thought as flattening n-cubes along a direction i.

A *cubical set* is a presheaf on $\square$. Intuitively, we can see a cubical set as an object constructed by a number of n-cubes (for arbitrary n), glued together in some way; such information is bundled together in a presheaf.

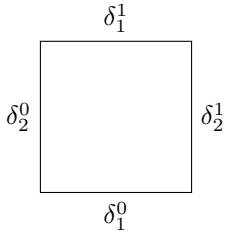
One of the examples of cubical sets are *standard n-cubes*, given by

$$\square^n = Hom_{\square}(-, I^n)$$

For example, $\square^2(I^k)$ is a collections of ways that k-cubes can be mapped to a square. For example

$$\square^2(I^2) = \{\delta_1^0(2), \delta_1^1(2), \delta_2^0(2), \delta_2^1(2)\}$$

which are four ways of mapping a line segment onto a square; this can be visualized as



A cubical set also tells us how to "glue" faces together. For example, consider $\square^2(\delta_1^1(0))(\delta_1^1) = \delta_1^1(1) \circ \delta_1^1(0)$ and $\square^2(\delta_1^1(0))(\delta_2^1(1)) = \delta_2^1(1) \circ \delta_1^1(0)$. Those two morphisms are identical:

$$(\delta_1^1(1) \circ \delta_1^1(0))(*) = (1, 1)$$
$$(\delta_1^2(1) \circ \delta_1^1(0))(*) = (1, 1)$$

That means that $\delta_2^1(1)$ and $\delta_1^1(1)$ has a common point (namely, $\delta_1^1(0)$) in $\square^2$, which can be visualized as in Figure 1.

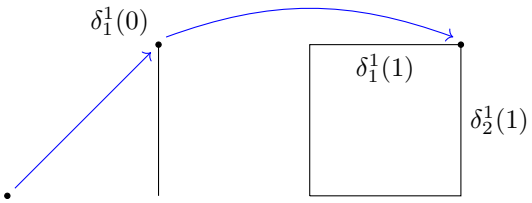


FIGURE 1. Visualization of $\delta_2^1(1)$ and $\delta_1^1(1)$ sharing a point

4.2.2. *Algebraic presentation.* Another presentation of cubical sets, used in Bezem et al. [2] and Huber [8], uses a dual category, formulated more algebraically; see [5, Remark 4.3] and [8, Remark 2.8].

Consider a countable set of variable names $\mathbb{A}$ (we usually reserve variables x, y, z for names), such that $0, 1 \notin \mathbb{A}$. Fix a category $\mathcal{C}$ finite subsets of $\mathbb{A}$ (we usually employ variables I, J, K for such sets). A morphism $f : I \to J$ in this category is a set-theoretic function $f : I \to J \cup \{2\}$, with the requirement that if $f(i) = f(j)$

and $f(i) \neq 0$ and $f(i) \neq 1$, then $i = j$; that is, f is injective on the *defined part*, which denoted by $def(f) := \{x \in I \mid f(x) \in J\}$.

We can think of a morphism $f : I \to J$ as a subsitution, which can only substitute in 0 or 1. There are two special kinds of maps in that category:

Face maps: Substitutions $(x = 0)$ and $(x = 1)$ going from I to $I - x$ (if x is in I); those are defined as

$$(x = 0) := y \mapsto \begin{cases} 0 \text{ if } y = x \\ y \text{ otherwise} \end{cases}$$

$$(x = 1) := y \mapsto \begin{cases} 1 \text{ if } y = x \\ y \text{ otherwise} \end{cases}$$

Degeneracy maps: Substitutions of the form $f : I \to J$, such that $f[I] \subseteq J$. A basic degeneracy map is a map of the form $\iota_x : I \to I \cup \{x\}$ for $x \notin I$.

The following lemma characterizes morphism in $\mathcal{C}$ and hints at why two presentations of cubical sets are equivalent:

Lemma 1 (Factoring lemma, see Huber [8, Lemma 2.2])**.** *Any morphism $f : I \to J$ can be written uniquely as $f = g \circ f_{01}$, where $f_{01} : I \to def(f)$ is a composition of face maps and $def(g) = def(f)$.*

A *cubical set* is then defined as a functor $\mathcal{C} \to \mathbf{Set}$ (that is, as a presheaf on $\mathcal{C}^{op}$). We denote that category of cubical sets as **cSet**. For each $X \in \mathbf{cSet}$ and $I \in \mathcal{C}$, we can think of an element $u \in X(I)$ as a hypercube of dimension $|I|$. We can obtain "faces" of u by considering $X(x = 0)(u), X(x = 1)(u) \in X(I - x)$.

The CwF structure on **cSet** is defined as in subsection 2.8. However, due to technical issues, such construction does not support non-trivial identity types. This is resolved by requiring additional structure on the cube category (of whichever variant), such as *filling conditions*, which over time has become increasingly complicated. See Bezem et al. [2] for details.

5. Going further

We have discussed a notion of models of type theory known as Categories with Families, which, in particular, is sound and complete. We have seen how this can be applied to show the independence of various claims, such as the unprovability of $0 \neq 1$ in the absence of universes, and the existence of nontrivial identity paths in the groupoid model.

There are other notions of models: one slightly modified variant is that of Categories with Attributes[5], where — unlike our assignment Tm of sets of terms, considering the motivation of subsection 2.3, we interpret the terms in a type to be a collection of sections: so we can go from a CwF to a CwA, simply by forgetting the definition of Tm.

We can essentially recover a CwF from a CwA by *defining* $\mathrm{Tm}(\Gamma, \sigma)$ to be that set of sections: but of course the exact terms we get are now, though comparable, not quite the original ones.

[5]The notion is originally due to Cartmell (unpublished); it is essentially the same as the notion of type-category by Pitts [11].

There are even more notions of models, such as *contextual categories* [12], which extend the definition of CwAs with a *length function*, and the more recent *natural models of homotopy type theory* by Awodey [1]. In most cases, the difference between the models is merely technical, and the various models are relatively easily translated into eachother.

Recent work on models of Homotopy Type Theory particularly revolves around that of *cubical sets* (see e.g. Bezem et al. [2], Licata and Brunerie [10] and Docherty [3]).

References

[1] Steve Awodey. Natural models of homotopy type theory (abstract). In Leonid Libkin, Ulrich Kohlenbach, and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation*, volume 8071 of *Lecture Notes in Computer Science*, pages 11–12. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39991-6. doi: 10.1007/978-3-642-39992-3_2. URL `http://dx.doi.org/10.1007/978-3-642-39992-3_2`.

[2] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In Ralph Matthes and Aleksy Schubert, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 107–128, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-72-9. doi: http://dx.doi.org/10.4230/LIPIcs.TYPES.2013.107. URL `http://drops.dagstuhl.de/opus/volltexte/2014/4628`.

[3] Simon Docherty. A model of type theory in cubical sets with connections. Master's thesis, University of Amsterdam, 2014.

[4] Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61780-8. doi: 10.1007/3-540-61780-9_66. URL `http://dx.doi.org/10.1007/3-540-61780-9_66`.

[5] Marco Grandis and Luca Mauri. Cubical sets and their site. *Theory and Applications of Categories*, 2003.

[6] Martin Hofmann. Syntax and semantics of dependent types. In P. Dybjer and A. Pitts, editors, *Extensional Constructs in Intensional Type Theory*, pages 13–54. Springer, 1997. ISBN 9780521118460.

[7] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.

[8] Simon Huber. *A Model of Type Theory in Cubical Sets*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, 7 2015. URL `http://www.cse.chalmers.se/~simonhu/misc/lic.pdf`.

[9] C. Kapulkin, P. LeFanu Lumsdaine, and V. Voevodsky. The Simplicial Model of Univalent Foundations. *ArXiv e-prints*, November 2012.

[10] Dan Licata and Guillaume Brunerie. A cubical approach to synthetic homotopy theory. *Preprint*, 2015.

[11] A. M. Pitts. Categorical logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*, chapter 2, pages 39–128. Oxford University Press, 2000. ISBN 0-19-853781-6. URL `http://www.oup.co.uk/isbn/0-19-853781-6`.

[12] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Birkhauser Boston Inc., Cambridge, MA, USA, 1991. ISBN 0-8176-3594-7.

[13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.