Type Theory

Lecture 3: Metatheory of Type Theory

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

SPLV Summer school, Edinburgh, 24 July 2025

https://fredriknf.com/splv2025/

Course plan

- **Monday:** Using type theory.
- **Tuesday:** Semantics of type theory.
- **Thursday:** ~~Implementation~~ Models, and metatheory.
  - Some concrete models, and what they are good for
  - Canonicity and normalisation

Slides and exercises: `https://fredriknf.com/splv2025/`

Reminder: categories with families

Definition A category with families (CwF) is given by:

- A category $\mathcal{C}$ with a terminal object.
- A presheaf $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$.
- A presheaf $\mathsf{Tm} : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$.
- A context extension $\Gamma \cdot A \in \mathcal{C}$ for every $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$ satisfying a certain universal property.

The Set model

We can take $\mathcal{C} = \mathbf{Set}$, the category of sets and functions.

We define $\mathsf{Ty}(\Gamma) :\equiv \Gamma \to \mathsf{Set}$.

Type substitution for $f : \Delta \to \Gamma$: $A[f] :\equiv A \circ f$.

We define $\mathsf{Tm}(\Gamma, A) :\equiv (\Pi\gamma \in \Gamma).A(\gamma)$.

Term substitution for $f : \Delta \to \Gamma$ and $t \in \mathsf{Tm}(\Gamma, A)$: $t[f]_\delta :\equiv t_{f(\delta)}$.

Finally we define $\Gamma \cdot A :\equiv (\Sigma\gamma \in \Gamma).A(\gamma)$ with $\mathsf{p} :\equiv \mathsf{fst}$, $\mathsf{q} :\equiv \mathsf{snd}$.

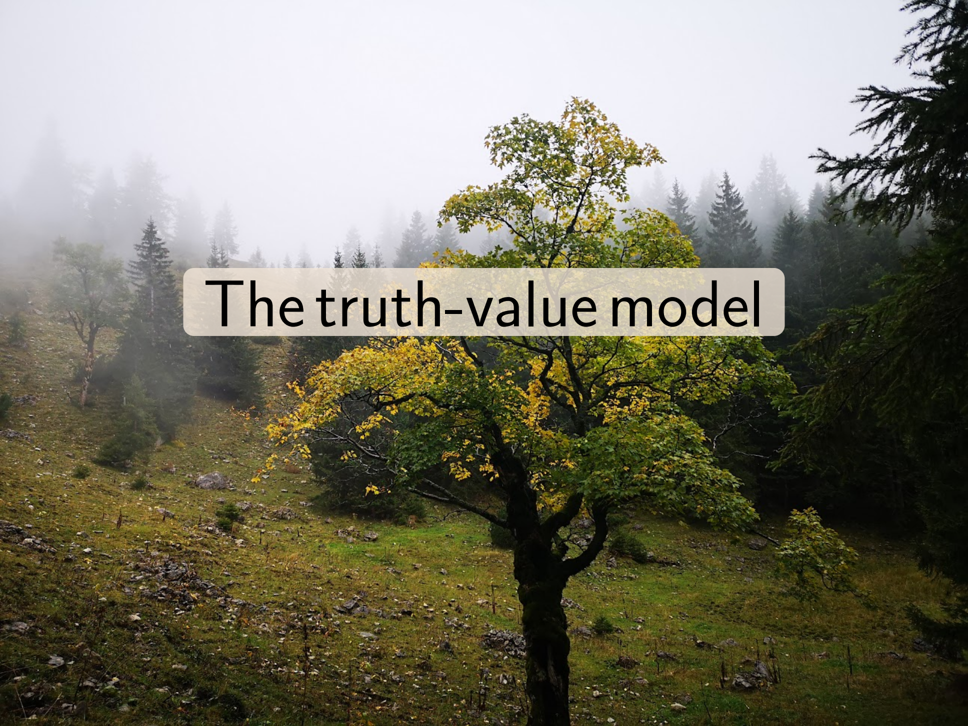
Some concrete models

Let us take a look at some concrete models and how they can be used for independence results:

- Smith's almost-trivial model (1988)
- Hofmann and Streicher's groupoid model (1994)
- A realizability model (see e.g. Beeson (1982))

Models such as the cubical sets model (Bezem, Coquand, and Huber 2013) can also inspire new syntax.

The truth-value model



Peano's Fourth Axiom

Using a universe, one can prove that $0 \neq \mathsf{suc}\, n$ for any $n : \mathbb{N}$.

Is it possible to prove this without using a universe?

Smith (1988) showed that this is impossible, by constructing a model where every type has at most one inhabitant.

The truth-value model

We take $\mathcal{C} := \{\mathsf{false}, \mathsf{true}\}$ with a unique morphism $\mathsf{false} \leq \mathsf{true}$.

The truth-value model

We take $\mathcal{C} := \{\mathsf{false}, \mathsf{true}\}$ with a unique morphism $\mathsf{false} \leq \mathsf{true}$.

We define

$$\mathsf{Ty}(\Gamma) := \{\mathsf{false}, \mathsf{true}\} \text{ for all (both) } \Gamma$$
$$A[\sigma] := A$$

The truth-value model

We take $\mathcal{C} := \{\text{false}, \text{true}\}$ with a unique morphism false $\leq$ true.

We define

$$\begin{aligned} \text{Ty}(\Gamma) &:= \{\text{false}, \text{true}\} \text{ for all (both) } \Gamma \\ A[\sigma] &:= A \end{aligned}$$

and

$$\begin{aligned} \text{Tm}(\Gamma, A) &:= \{\star \mid \Gamma \leq A\} \\ t[\sigma] &:= t \end{aligned}$$

That is, there is a (unique) term of type A unless $\Gamma = \text{true}$ and $A = \text{false}$.

The truth-value model

We take $\mathcal{C} := \{\text{false}, \text{true}\}$ with a unique morphism false $\leq$ true.

We define

$$\begin{aligned} \text{Ty}(\Gamma) &:= \{\text{false}, \text{true}\} \text{ for all (both) } \Gamma \\ A[\sigma] &:= A \end{aligned}$$

and

$$\begin{aligned} \text{Tm}(\Gamma, A) &:= \{\star \mid \Gamma \leq A\} \\ t[\sigma] &:= t \end{aligned}$$

That is, there is a (unique) term of type A unless $\Gamma = \text{true}$ and $A = \text{false}$.

We take $\Gamma \cdot A := \Gamma \wedge A$, for which we can define $\mathsf{p} : \Gamma \wedge A \leq \Gamma$ and $\mathsf{q} = \star \in \text{Tm}(\Gamma \wedge A, A)$.

Interpreting the type formers

The plan is to interpret potentially inhabited types as true and empty types as false.

Interpreting the type formers

The plan is to interpret potentially inhabited types as true and empty types as false.

Hence we define

$$\begin{aligned}
\mathsf{Empty} &:= \mathsf{false} \\
\mathsf{Unit} &:= \mathsf{true} \\
\mathsf{Nat} &:= \mathsf{true} \\
\Pi\, A\, B &:= A \supset B \qquad \text{(Boolean implication)} \\
\Sigma\, A\, B &:= A \wedge B \\
\mathsf{Id}(A, a, b) &:= \mathsf{true}
\end{aligned}$$

Interpreting the type formers

The plan is to interpret potentially inhabited types as true and empty types as false.

Hence we define

$$
\begin{aligned}
\mathsf{Empty} &:= \mathsf{false} \\
\mathsf{Unit} &:= \mathsf{true} \\
\mathsf{Nat} &:= \mathsf{true} \\
\Pi\, A\, B &:= A \supset B \qquad \text{(Boolean implication)} \\
\Sigma\, A\, B &:= A \wedge B \\
\mathsf{Id}(A, a, b) &:= \mathsf{true}
\end{aligned}
$$

Whenever we are asked to interpret a term, we can use $\star$ by construction.

$0 \neq \mathsf{suc}\, n$ in the model?

What are the terms of type $(0 = \mathsf{suc}\, n) \to \mathbf{0}$ in the model?

$0 \neq \mathsf{suc}\, n$ in the model?

What are the terms of type $(0 = \mathsf{suc}\, n) \to \mathbf{0}$ in the model?

$$\begin{aligned}\mathsf{Tm}(1, \mathsf{Id}(\mathsf{Nat}, 0, \mathsf{suc}\, n) \to \mathsf{Empty}) &= \mathsf{Tm}(\mathsf{true}, \mathsf{true} \supset \mathsf{false}) \\ &= \{\star \mid \mathsf{true} \leq \mathsf{false}\} \\ &= \emptyset\end{aligned}$$

$0 \neq \mathsf{suc}\, n$ in the model?

What are the terms of type $(0 = \mathsf{suc}\, n) \to \mathbf{0}$ in the model?

$$\begin{aligned} \mathsf{Tm}(1, \mathsf{Id}(\mathsf{Nat}, 0, \mathsf{suc}\, n) \to \mathsf{Empty}) &= \mathsf{Tm}(\mathsf{true}, \mathsf{true} \supset \mathsf{false}) \\ &= \{\star \mid \mathsf{true} \leq \mathsf{false}\} \\ &= \emptyset \end{aligned}$$

Hence by soundness, there cannot be a proof of $(0 = \mathsf{suc}\, n) \to \mathbf{0}$, since such a proof would be interpreted by an element of $\emptyset$.

$0 \neq \mathsf{suc}\, n$ in the model?

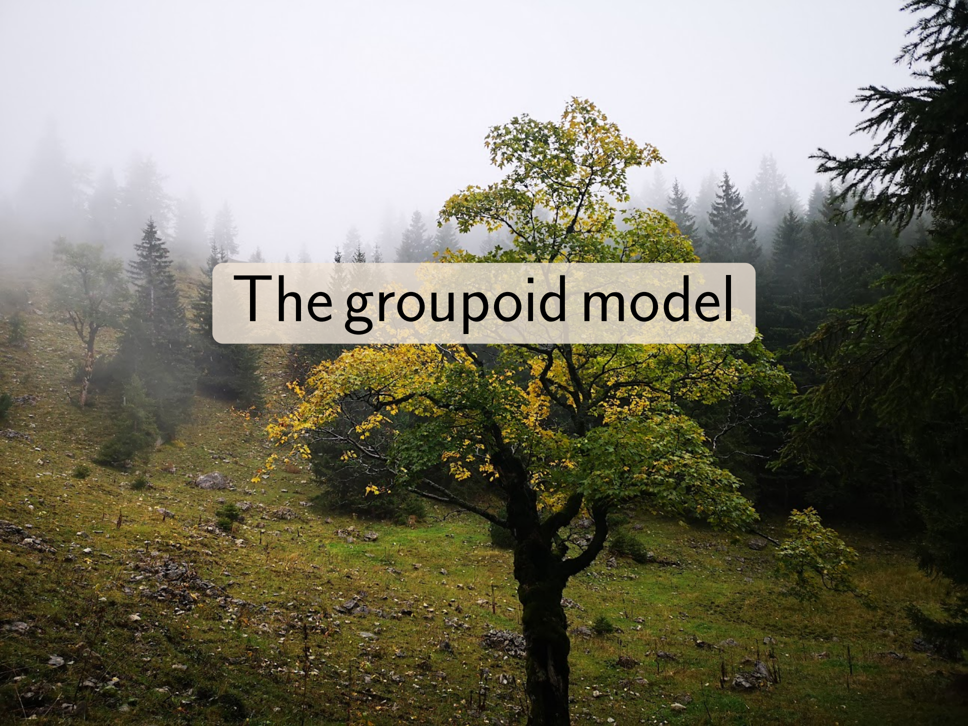
What are the terms of type $(0 = \mathsf{suc}\, n) \to \mathbf{0}$ in the model?

$$\begin{aligned}
\mathsf{Tm}(1, \mathsf{Id}(\mathsf{Nat}, 0, \mathsf{suc}\, n) \to \mathsf{Empty}) &= \mathsf{Tm}(\mathsf{true}, \mathsf{true} \supset \mathsf{false}) \\
&= \{\star \mid \mathsf{true} \leq \mathsf{false}\} \\
&= \emptyset
\end{aligned}$$

Hence by soundness, there cannot be a proof of $(0 = \mathsf{suc}\, n) \to \mathbf{0}$, since such a proof would be interpreted by an element of $\emptyset$.

Note The model does not support universes, because they cannot afford to ignore all dependencies!

The groupoid model



Uniqueness of identity proofs?

Given $p, q : a =_A b$, is it possible to prove $p =_{a =_A b} q$?

Uniqueness of identity proofs?

Given $p, q : a =_A b$, is it possible to prove $p =_{a =_A b} q$?

This is true in the **Set** model (so we cannot hope to disprove it).

Uniqueness of identity proofs?

Given $p, q : a =_A b$, is it possible to prove $p =_{a=_A b} q$?

This is true in the **Set** model (so we cannot hope to disprove it).

Also provable in a natural extension of type theory: Streicher's Axiom K (1993) or Coquand's dependent pattern matching (1992). (Mc Bride (1999) showed that in fact Axiom K and pattern matching are equivalent.)

Identities between identity proofs

Some equations are provable:

$$\begin{aligned}
\mathrm{trans}(p, \mathrm{refl}) &= p \\
\mathrm{trans}(\mathrm{refl}, q) &= q \\
\mathrm{trans}(\mathrm{trans}(p, q), r) &= \mathrm{trans}(p, \mathrm{trans}(q, r)) \\
\mathrm{trans}(p, \mathrm{sym}(p)) &= \mathrm{refl} \\
\mathrm{trans}(\mathrm{sym}(q), q) &= q
\end{aligned}$$

Identities between identity proofs

Some equations are provable:

$$\begin{aligned}
\mathsf{trans}(p, \mathsf{refl}) &= p \\
\mathsf{trans}(\mathsf{refl}, q) &= q \\
\mathsf{trans}(\mathsf{trans}(p, q), r) &= \mathsf{trans}(p, \mathsf{trans}(q, r)) \\
\mathsf{trans}(p, \mathsf{sym}(p)) &= \mathsf{refl} \\
\mathsf{trans}(\mathsf{sym}(q), q) &= q
\end{aligned}$$

So identity types makes every type into a groupoid — at least up to higher identity types! $\rightsquigarrow$ ∞-groupoids.

Identities between identity proofs

Some equations are provable:

$$\begin{aligned} \mathrm{trans}(p, \mathrm{refl}) &= p \\ \mathrm{trans}(\mathrm{refl}, q) &= q \\ \mathrm{trans}(\mathrm{trans}(p, q), r) &= \mathrm{trans}(p, \mathrm{trans}(q, r)) \\ \mathrm{trans}(p, \mathrm{sym}(p)) &= \mathrm{refl} \\ \mathrm{trans}(\mathrm{sym}(q), q) &= q \end{aligned}$$

So identity types makes every type into a groupoid — at least up to higher identity types! $\rightsquigarrow$ ∞-groupoids.

Further, every function respects equality, so from this perspective, every function is a functor between groupoids, etc.

Identities between identity proofs

Some equations are provable:

$$\begin{aligned}
\mathsf{trans}(p, \mathsf{refl}) &= p \\
\mathsf{trans}(\mathsf{refl}, q) &= q \\
\mathsf{trans}(\mathsf{trans}(p, q), r) &= \mathsf{trans}(p, \mathsf{trans}(q, r)) \\
\mathsf{trans}(p, \mathsf{sym}(p)) &= \mathsf{refl} \\
\mathsf{trans}(\mathsf{sym}(q), q) &= q
\end{aligned}$$

So identity types makes every type into a groupoid — at least up to higher identity types! $\rightsquigarrow$ ∞-groupoids.

Further, every function respects equality, so from this perspective, every function is a functor between groupoids, etc.

Hofmann's insight: we can turn this around and make a model out of groupoids!

The groupoid model

We take $\mathcal{C} := \mathbf{Gpd}$, the category of groupoids and functors.

The groupoid model

We take $\mathcal{C} := \mathbf{Gpd}$, the category of groupoids and functors.

We define $\mathsf{Ty}(\Gamma) := [\Gamma, \mathbf{Gpd}]$ (functors from Γ to $\mathbf{Gpd}$)

The groupoid model

We take $\mathcal{C} \coloneqq \mathbf{Gpd}$, the category of groupoids and functors.

We define $\mathrm{Ty}(\Gamma) \coloneqq [\Gamma, \mathbf{Gpd}]$ (functors from Γ to $\mathbf{Gpd}$)

If $f : \Delta \to \Gamma$, we can take $A[f] \coloneqq A \circ f : [\Delta, \mathbf{Gpd}]$.

The groupoid model

We take $\mathcal{C} \coloneqq \mathbf{Gpd}$, the category of groupoids and functors.

We define $\mathrm{Ty}(\Gamma) \coloneqq [\Gamma, \mathbf{Gpd}]$ (functors from Γ to $\mathbf{Gpd}$)

If $f : \Delta \to \Gamma$, we can take $A[f] \coloneqq A \circ f : [\Delta, \mathbf{Gpd}]$.

Terms $\mathrm{Tm}(\Gamma, A)$ are "dependent functors":

$$M_0 \in (\Pi\gamma \in \Gamma).A(\gamma)$$
$$M_1 \in (\Pi f : \gamma \to \gamma').\big(A(f)(M_0(\gamma)) \to M_0(\gamma')\big)$$

s.t. $M_1(\mathrm{id}_\gamma) = \mathrm{id}_{M_0(\gamma)}$ and $M_1(f \circ g) = M_1(f) \circ A(f)(M_1(g))$.
Substitution is again composition.

The groupoid model

We take $\mathcal{C} \coloneqq \mathbf{Gpd}$, the category of groupoids and functors.

We define $\mathrm{Ty}(\Gamma) \coloneqq [\Gamma, \mathbf{Gpd}]$ (functors from Γ to $\mathbf{Gpd}$)

If $f : \Delta \to \Gamma$, we can take $A[f] \coloneqq A \circ f : [\Delta, \mathbf{Gpd}]$.

Terms $\mathrm{Tm}(\Gamma, A)$ are "dependent functors":

$$M_0 \in (\Pi\gamma \in \Gamma).A(\gamma)$$
$$M_1 \in (\Pi f : \gamma \to \gamma').\big(A(f)(M_0(\gamma)) \to M_0(\gamma')\big)$$

s.t. $M_1(\mathrm{id}_\gamma) = \mathrm{id}_{M_0(\gamma)}$ and $M_1(f \circ g) = M_1(f) \circ A(f)(M_1(g))$.
Substitution is again composition.

We define $\Gamma \cdot A \coloneqq \int_\Gamma A$, i.e., objects are pairs $(\gamma \in \Gamma, a \in A(\gamma))$ and $(f, g) : (\gamma, a) \to (\gamma', a')$ if $f : \gamma \to \gamma'$ and $g : A(f)(a) \to a'$.

Interpreting identity types

We interpret $\mathsf{Id}\, A\, a\, b$ as the discrete groupoid with objects $\mathrm{Hom}_A(a, b)$. On morphisms, we define $(\mathsf{Id}\, A\, f\, g)(r) :\equiv g \circ r \circ f^{-1}$.

Interpreting identity types

We interpret $\mathsf{Id}\,A\,a\,b$ as the discrete groupoid with objects $\mathsf{Hom}_A(a, b)$. On morphisms, we define $(\mathsf{Id}\,A\,f\,g)(r) :\equiv g \circ r \circ f^{-1}$.

For $\mathsf{refl} : \mathsf{Id}\,A\,a\,a$, we can take $\mathsf{refl} :\equiv \mathsf{id}_a \in \mathsf{Hom}_A(a, a)$.

Interpreting identity types

We interpret $\mathsf{Id}\,A\,a\,b$ as the discrete groupoid with objects $\mathrm{Hom}_A(a, b)$. On morphisms, we define $(\mathsf{Id}\,A\,f\,g)(r) :\equiv g \circ r \circ f^{-1}$.

For $\mathsf{refl} : \mathsf{Id}\,A\,a\,a$, we can take $\mathsf{refl} :\equiv \mathrm{id}_a \in \mathrm{Hom}_A(a, a)$.

For $\mathsf{elim}_=$, we are given $d(x) \in C(x, x, \mathrm{id}_x)$ and $r : \mathsf{Id}(x, y)$, and must construct $\mathsf{elim}_=(d, r) \in C(x, y, r)$.

Interpreting identity types

We interpret $\mathrm{Id}\,A\,a\,b$ as the discrete groupoid with objects $\mathrm{Hom}_A(a, b)$. On morphisms, we define $(\mathrm{Id}\,A\,f\,g)(r) := g \circ r \circ f^{-1}$.

For refl : $\mathrm{Id}\,A\,a\,a$, we can take refl $:= \mathrm{id}_a \in \mathrm{Hom}_A(a, a)$.

For $\mathrm{elim}_=$, we are given $d(x) \in C(x, x, \mathrm{id}_x)$ and $r : \mathrm{Id}(x, y)$, and must construct $\mathrm{elim}_=(d, r) \in C(x, y, r)$.

C is a functor, so it suffices to construct a morphism $(x, x, \mathrm{id}_x) \to (x, y, r)$. Such a morphism is given by

$$
\begin{aligned}
f &: x \to x && \text{in } A \\
g &: x \to y && \text{in } A \\
h &: \mathrm{Id}_A(f, g)(\mathrm{id}_x) \to r && \text{in } \mathrm{Id}_A(x, y)
\end{aligned}
$$

Interpreting identity types

We interpret $\mathsf{Id}\,A\,a\,b$ as the discrete groupoid with objects $\mathsf{Hom}_A(a,b)$. On morphisms, we define $(\mathsf{Id}\,A\,f\,g)(r) :\equiv g \circ r \circ f^{-1}$.

For $\mathsf{refl} : \mathsf{Id}\,A\,a\,a$, we can take $\mathsf{refl} :\equiv \mathsf{id}_a \in \mathsf{Hom}_A(a,a)$.

For $\mathsf{elim}_=$, we are given $d(x) \in C(x,x,\mathsf{id}_x)$ and $r : \mathsf{Id}(x,y)$, and must construct $\mathsf{elim}_=(d,r) \in C(x,y,r)$.

C is a functor, so it suffices to construct a morphism $(x,x,\mathsf{id}_x) \to (x,y,r)$. Such a morphism is given by

$$
\begin{aligned}
f &: x \to x && \text{in } A \\
g &: x \to y && \text{in } A \\
h &: f^{-1} \circ \mathsf{id}_x \circ g = r && \text{in } \mathsf{Id}_A(x,y)
\end{aligned}
$$

Interpreting identity types

We interpret $\mathrm{Id}\,A\,a\,b$ as the discrete groupoid with objects $\mathrm{Hom}_A(a, b)$. On morphisms, we define $(\mathrm{Id}\,A\,f\,g)(r) :\equiv g \circ r \circ f^{-1}$.

For $\mathrm{refl} : \mathrm{Id}\,A\,a\,a$, we can take $\mathrm{refl} :\equiv \mathrm{id}_a \in \mathrm{Hom}_A(a, a)$.

For $\mathrm{elim}_=$, we are given $d(x) \in C(x, x, \mathrm{id}_x)$ and $r : \mathrm{Id}(x, y)$, and must construct $\mathrm{elim}_=(d, r) \in C(x, y, r)$.

C is a functor, so it suffices to construct a morphism $(x, x, \mathrm{id}_x) \to (x, y, r)$. Such a morphism is given by

$$
\begin{aligned}
f &: x \to x && \text{in } A \\
g &: x \to y && \text{in } A \\
h &: f^{-1} \circ \mathrm{id}_x \circ g = r && \text{in } \mathrm{Id}_A(x, y)
\end{aligned}
$$

So we can take $f = \mathrm{id}$, $g = r$, $h = \mathrm{id}$, and define $\mathrm{elim}_=(d, r) :\equiv C(\mathrm{id}, r, \mathrm{id})(d(x))$. (We also need to define actions on morphisms.)

Interpreting other type formers

Other type fomers can be interpreted much as in the **Set** model, after taking care to define actions on morphisms.

Interpreting other type formers

Other type fomers can be interpreted much as in the **Set** model, after taking care to define actions on morphisms.

Of particular interest is the universe.

Interpreting other type formers

Other type fomers can be interpreted much as in the **Set** model, after taking care to define actions on morphisms.

Of particular interest is the universe.

Given a set-theoretic universe V, we define $U : \Gamma \to \mathbf{Gpd}$ as $U(\gamma) := \mathsf{Gpd}_V$, the groupoid of V-small groupoids, with an inclusion $\mathsf{El} : \mathsf{Gpd}_V \hookrightarrow \mathbf{Gpd}$.

Interpreting other type formers

Other type fomers can be interpreted much as in the **Set** model, after taking care to define actions on morphisms.

Of particular interest is the universe.

Given a set-theoretic universe V, we define $U : \Gamma \to \mathbf{Gpd}$ as $U(\gamma) := \mathsf{Gpd}_V$, the groupoid of V-small groupoids, with an inclusion $\mathsf{El} : \mathsf{Gpd}_V \hookrightarrow \mathbf{Gpd}$.

That is: the objects of Gpd_V are groupoids whose object set and morphism sets live in V, and the morphisms in Gpd_V are isomorphisms.

Interpreting other type formers

Other type fomers can be interpreted much as in the **Set** model, after taking care to define actions on morphisms.

Of particular interest is the universe.

Given a set-theoretic universe V, we define $U : \Gamma \to \mathbf{Gpd}$ as $U(\gamma) := \mathrm{Gpd}_V$, the groupoid of V-small groupoids, with an inclusion $\mathrm{El} : \mathrm{Gpd}_V \hookrightarrow \mathbf{Gpd}$.

That is: the objects of Gpd_V are groupoids whose object set and morphism sets live in V, and the morphisms in Gpd_V are isomorphisms.

In particular, this means that $A =_U B$ in the model iff $A \cong B$.
$\rightsquigarrow$ Precursor to the Univalence Axiom.

Refuting UIP

Let G be your favourite non-trivial group (e.g. $G = (\mathbb{Z}, +, 0)$) and consider the one-element groupoid BG with $BG(\star, \star) = G$.

Refuting UIP

Let G be your favourite non-trivial group (e.g. $G = (\mathbb{Z}, +, 0)$) and consider the one-element groupoid BG with $BG(\star, \star) = G$.

Suppose u is a proof of UIP, i.e.,

$$u : (\Pi A : U)(\Pi a : \mathsf{El}(A))(\Pi b : \mathsf{El}(A))(\Pi p : a = b))(\Pi q : a = b)).p = q$$

Refuting UIP

Let G be your favourite non-trivial group (e.g. $G = (\mathbb{Z}, +, 0)$) and consider the one-element groupoid BG with $BG(\star, \star) = G$.

Suppose u is a proof of UIP, i.e.,

$u : (\Pi A : U)(\Pi a : \mathsf{El}(A))(\Pi b : \mathsf{El}(A))(\Pi p : a = b))(\Pi q : a = b)).p = q$

We would then have

$$u(B\mathbb{Z}, \star, \star, 0, 1) \in \mathsf{Id}\,(\mathsf{Id}\,B\mathbb{Z}\;\star\;\star)\,0\,1$$

in the model, but $\mathsf{Id}\,B\mathbb{Z}\;\star\;\star$ is a discrete groupoid, hence $\mathsf{Id}\,(\mathsf{Id}\,B\mathbb{Z}\;\star\;\star)\,0\,1 = \emptyset$ since $0 \neq 1$. Hence no such proof u can exist.

Going higher

Because each $\mathsf{Id}\, A\, a\, b$ is discrete, the model does validate uniqueness of identity proofs between identity proofs ("UIPIP").

Going higher

Because each Id $A\,a\,b$ is discrete, the model does validate uniqueness of identity proofs between identity proofs ("UIPIP").

Their uniqueness can be refuted in a model of 2-groupoids, then we might want to move to 3-groupoids to refute UIPIPIP, etc.

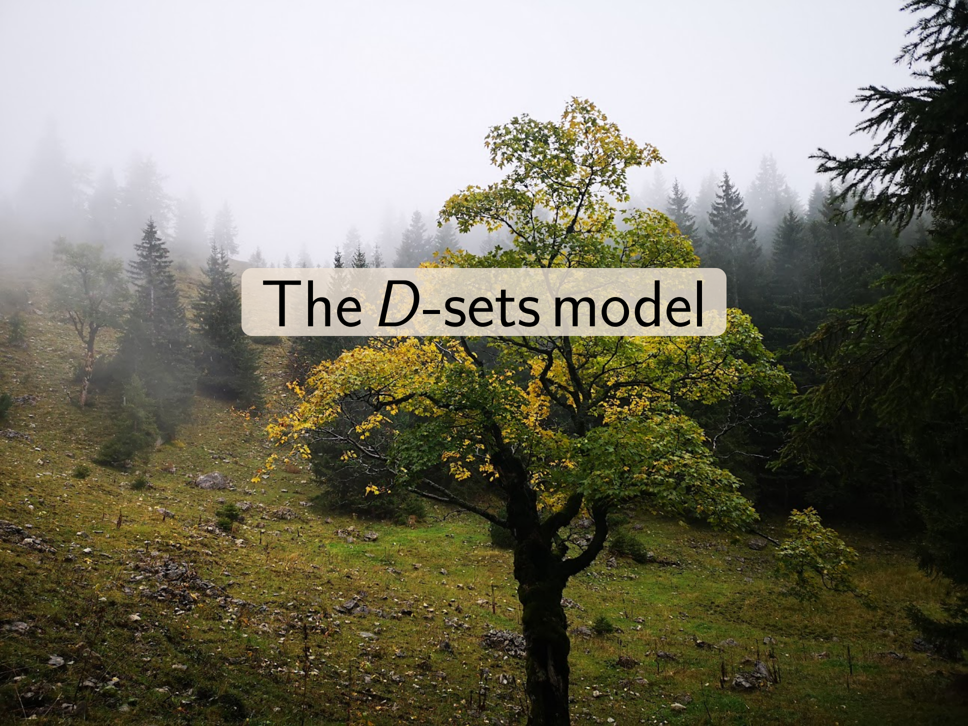
Going higher

Because each $\mathsf{Id}\, A\, a\, b$ is discrete, the model does validate uniqueness of identity proofs between identity proofs ("UIPIP").

Their uniqueness can be refuted in a model of 2-groupoids, then we might want to move to 3-groupoids to refute UIPIPIP, etc.

In the limit, we would rediscover Voevodsky's simplicial sets (aka ∞-groupoids) model of homotopy type theory (Kapulkin and Lumsdaine, 2021).

The D-sets model



A model based on computation

Intuitively, all constructions of type theory are computable. Can we make this precise?

A model based on computation

Intuitively, all constructions of type theory are computable. Can we make this precise?

We will construct a model where each term has an associated piece of "computation data" from a *model of computation D*.

A model based on computation

Intuitively, all constructions of type theory are computable. Can we make this precise?

We will construct a model where each term has an associated piece of "computation data" from a *model of computation* D.

Definition A combinatory algebra is a set D with a binary operation $\$: D \times D \to D$ together with elements $K, S \in D$ such that

$$K \$ x \$ y = x \qquad\qquad S \$ x \$ y \$ z = (x \$ z) \$ (y \$ z)$$

(Can also work with *partial* combinatory algebras, i.e. $\$$ partial.)

Examples $D =$ untyped lambda terms, $D =$ an enumeration of Turing machines as natural numbers.

Functional completeness

D is functionally complete: for each term $t(x_1, \ldots, x_n) \in D$ there is $f \in D$ such that $f \mathbin{\$} a_1 \mathbin{\$} \ldots \mathbin{\$} a_n = t(a_1, \ldots, a_n)$.

Functional completeness

D is functionally complete: for each term $t(x_1, \ldots, x_n) \in D$ there is $f \in D$ such that $f \mathbin{\$} a_1 \mathbin{\$} \ldots \mathbin{\$} a_n = t(a_1, \ldots, a_n)$.

Hence we can do the usual Church encoding tricks and define pairing and projections: There are $\pi_1, \pi_2, <a, b> \in D$ such that

$$\pi_1 \mathbin{\$} a \mathbin{\$} b = a$$
$$\pi_2 \mathbin{\$} a \mathbin{\$} b = b$$
$$<a, b> \mathbin{\$} c = c \mathbin{\$} a \mathbin{\$} b$$

Hence $<a, b> \mathbin{\$} \pi_1 = a$ and $<a, b> \mathbin{\$} \pi_2 = b$.

Functional completeness

D is functionally complete: for each term $t(x_1, \ldots, x_n) \in D$ there is $f \in D$ such that $f \$ a_1 \$ \ldots \$ a_n = t(a_1, \ldots, a_n)$.

Hence we can do the usual Church encoding tricks and define pairing and projections: There are $\pi_1, \pi_2, <a, b> \in D$ such that

$$\pi_1 \$ a \$ b = a$$
$$\pi_2 \$ a \$ b = b$$
$$<a, b> \$ c = c \$ a \$ b$$

Hence $<a, b> \$ \pi_1 = a$ and $<a, b> \$ \pi_2 = b$.

Similarly we can define Church numerals c_n for natural numbers.

D-sets and their morphisms

Definition A D-set (or assembly) is a pair $(X, \Vdash_X)$, where X is a set and $\Vdash_X \subseteq D \times X$, such that for each $x \in X$, there exists $a \in D$ such that $a \Vdash_X x$.

D-sets and their morphisms

Definition A D-set (or assembly) is a pair $(X, \Vdash_X)$, where X is a set and $\Vdash_X \subseteq D \times X$, such that for each $x \in X$, there exists $a \in D$ such that $a \Vdash_X x$.

A morphism $(X, \Vdash_X) \to (Y, \Vdash_Y)$ is a function $X \to Y$ such that there exists $d \in D$ such that if $a \Vdash_X x$ then $d \, \$ \, a \Vdash_Y f(x)$.

D-sets and their morphisms

Definition A D-set (or assembly) is a pair $(X, \Vdash_X)$, where X is a set and $\Vdash_X \subseteq D \times X$, such that for each $x \in X$, there exists $a \in D$ such that $a \Vdash_X x$.

A morphism $(X, \Vdash_X) \to (Y, \Vdash_Y)$ is a function $X \to Y$ such that there exists $d \in D$ such that if $a \Vdash_X x$ then $d \,\$\, a \Vdash_Y f(x)$.

Terminology if $a \Vdash_X x$, we call a a realizer of x. We say that d above tracks f.

D-sets and their morphisms

Definition A D-set (or assembly) is a pair $(X, \Vdash_X)$, where X is a set and $\Vdash_X \subseteq D \times X$, such that for each $x \in X$, there exists $a \in D$ such that $a \Vdash_X x$.

A morphism $(X, \Vdash_X) \to (Y, \Vdash_Y)$ is a function $X \to Y$ such that there exists $d \in D$ such that if $a \Vdash_X x$ then $d \,\$\, a \Vdash_Y f(x)$.

Terminology if $a \Vdash_X x$, we call a a realizer of x. We say that d above tracks f.

There is an identity morphism, and D-set morphisms compose (easy by functional completeness).

The category of D-sets

The category of D-sets has lots of nice structure:

- Products $(X, \Vdash_X) \times (Y, \Vdash_Y) = (X \times Y, \Vdash)$ where $d \Vdash (x, y)$ iff $d = <a, b>$ such that $a \Vdash_X x$ and $b \Vdash_Y y$.
- Exponentials $(X, \Vdash_X) \Rightarrow (Y, \Vdash_Y)$ with underlying sets D-sets morphisms, and $d \Vdash f$ iff d tracks f.
- A natural numbers objects $(\mathbb{N}, \Vdash_{\mathbb{N}})$ where $d \Vdash_{\mathbb{N}} n$ iff $d = c_n$.
- Coproducts $(X_0, \Vdash_{X_0}) + (X_1, \Vdash_{X_1}) = (X_0 + X_1, \Vdash)$ where $d \Vdash \mathrm{in}_i\, x$ iff $d = <c_i, a>$ such that $a \Vdash_{X_i} x$.

D-sets as a CwF

We build a category with families structure on the category of D-sets.

We take

$$\begin{aligned} \mathrm{Ty}((X, \Vdash_X)) &:= X \to D\text{-Set} \\ \mathrm{Tm}((X, \Vdash_X), Y) &:= \{b : (\Pi x \in X).Y(x) \mid \exists d \in D.d \text{ tracks } b\} \end{aligned}$$

and define $(X, \Vdash_X) \cdot Y := ((\Sigma x \in X).Y(x), \Vdash)$ where $d \Vdash (x, y)$ iff $d = <a, b>$ such that $a \Vdash_X x$ and $b \Vdash_{Y(x)} y$.

D-sets as a CwF

We build a category with families structure on the category of D-sets.

We take

$$\mathrm{Ty}((X, \Vdash_X)) \coloneqq X \to D\text{-Set}$$
$$\mathrm{Tm}((X, \Vdash_X), Y) \coloneqq \{b : (\Pi x \in X).Y(x) \mid \exists d \in D.d \text{ tracks } b\}$$

and define $(X, \Vdash_X) \cdot Y \coloneqq ((\Sigma x \in X).Y(x), \Vdash)$ where $d \Vdash (x, y)$ iff $d = <a, b>$ such that $a \Vdash_X x$ and $b \Vdash_{Y(x)} y$.

Using the categorical structure in D-Set, we interpret (dependent) functions and pairs, disjoint unions, natural numbers, etc.

An impredicative universe

There is an interesting subcategory of so-called modest D-sets:

Definition A D-set $(X, \Vdash_X)$ is modest if $d \Vdash_X x$ and $d \Vdash_X y$ implies $x = y$. (A family $Y : X \to D$-Set is called modest if each Y_x is modest.)

An impredicative universe

There is an interesting subcategory of so-called modest D-sets:

Definition A D-set $(X, \Vdash_X)$ is modest if $d \Vdash_X x$ and $d \Vdash_X y$ implies $x = y$. (A family $Y : X \to D$-Set is called modest if each Y_x is modest.)

Example Unless D is trivial, $(\mathbb{N}, \Vdash_{\mathbb{N}})$ is modest.

An impredicative universe

There is an interesting subcategory of so-called modest D-sets:

Definition A D-set $(X, \Vdash_X)$ is modest if $d \Vdash_X x$ and $d \Vdash_X y$ implies $x = y$. (A family $Y : X \to D$-Set is called modest if each Y_x is modest.)

Example Unless D is trivial, $(\mathbb{N}, \Vdash_{\mathbb{N}})$ is modest.

Modest sets are isomorphic to partial equivalence relations on D, hence "all small". Thus: if $B \in \mathrm{Ty}(\Gamma \cdot A)$ is modest then $\Pi\, A\, B \in \mathrm{Ty}(\Gamma)$ is modest, for all $A \in \mathrm{Ty}(\Gamma)$.

An impredicative universe

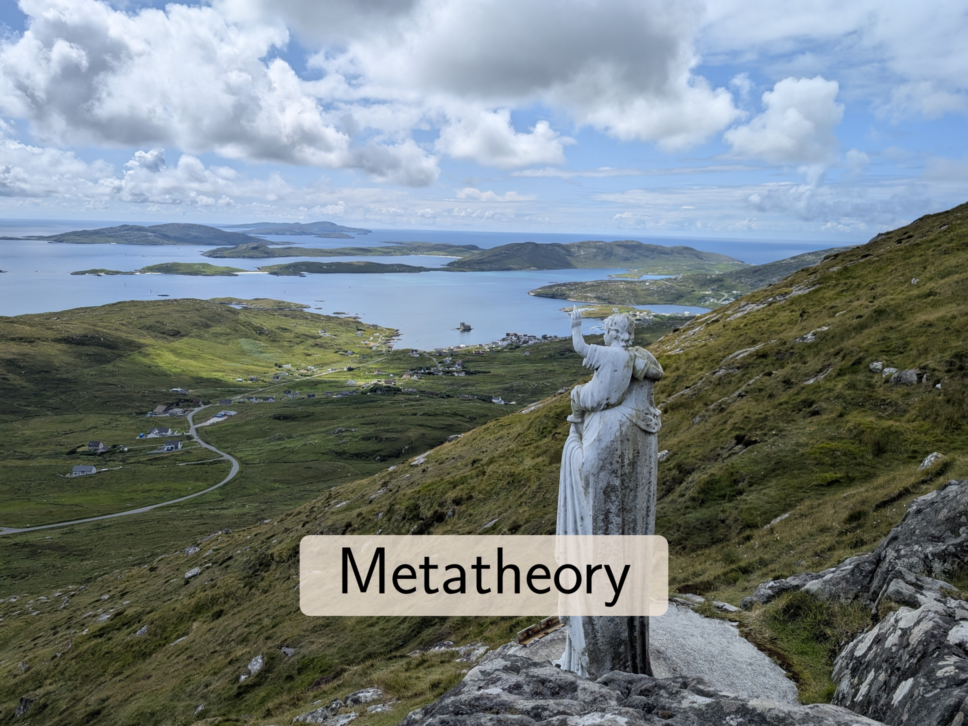
There is an interesting subcategory of so-called modest D-sets:

Definition A D-set $(X, \Vdash_X)$ is modest if $d \Vdash_X x$ and $d \Vdash_X y$ implies $x = y$. (A family $Y : X \to D$-Set is called modest if each Y_x is modest.)

Example Unless D is trivial, $(\mathbb{N}, \Vdash_{\mathbb{N}})$ is modest.

Modest sets are isomorphic to partial equivalence relations on D, hence "all small". Thus: if $B \in \mathrm{Ty}(\Gamma \cdot A)$ is modest then $\Pi\, A\, B \in \mathrm{Ty}(\Gamma)$ is modest, for all $A \in \mathrm{Ty}(\Gamma)$.

Modest sets form a universe closed under impredicative quantification, containing the natural numbers. Such a universe contradicts classical logic.


Metatheory

Metatheory

What kind of properties of a type theory might one care about?

Metatheory

What kind of properties of a type theory might one care about?

- **Consistency:** There is no proof of $\mathbf{0}$ in the empty context.
- **Canonicity:** Every closed term of type $\mathbb{N}$ is equal to a numeral $\mathsf{suc}^n\,0$.
- **Normalisation:** Every term is equal to a term in *normal form*.
- (**Strong normalisation:** Every term reduces to a term in normal form, no matter the reduction strategy.)

Consistency

Exhibited by (e.g.) the **Set** model:

If there was a proof of $\mathbf{0}$, it would be interpreted as an element of $\emptyset$ in the **Set** model, which is absurd.

Consistency

Exhibited by (e.g.) the **Set** model:

If there was a proof of $\mathbf{0}$, it would be interpreted as an element of $\emptyset$ in the **Set** model, which is absurd.

If you propose an extension to a type theory, you want to know/show that it is still consistent. But there is not much you can do with a proof of consistency.

Canonicity

Canonicity: Every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\, 0$.

Canonicity

Canonicity: Every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\,0$.

The same in spirit: "Every closed term of type Bool is (judgementally) equal to true or false".

Canonicity

Canonicity: Every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\,0$.

The same in spirit: "Every closed term of type Bool is (judgementally) equal to true or false".

How do we prove it? Unfortunately, a naive induction on typing judgements does not work.

A "proof-relevant" logical relation (Coquand 2019)

To each (closed) type A we associate a family of *sets* $A' : A \to \mathsf{Set}$ of "proofs of canonicity".

To each closed term $t : A$, we associate an element $t' \in A'(t)$.

A “proof-relevant” logical relation (Coquand 2019)

To each (closed) type A we associate a family of *sets* $A' : A \to \mathrm{Set}$ of “proofs of canonicity”.

To each closed term $t : A$, we associate an element $t' \in A'(t)$.

$$\begin{aligned} \mathbb{N}'(t) &\coloneqq \{n \mid t \equiv \mathrm{suc}^n\, 0\} \\ \big((\Pi x : A).B\big)'(t) &\coloneqq (\Pi a : A)(\Pi a' : A'(a)).B'(a, a')\,(t\, a) \end{aligned}$$

$$\begin{aligned} (t\, a)' &\coloneqq t'\, a\, a' \\ \big((\lambda x : A).t\big)' &\coloneqq (\lambda a : A)(\lambda a' : A'(a)).t'\, a\, a' \\ (\mathrm{suc}\, n)' &\coloneqq n' + 1 \\ &\vdots \end{aligned}$$

A "proof-relevant" logical relation (Coquand 2019)

To each (closed) type A we associate a family of *sets* $A' : A \to \mathsf{Set}$ of "proofs of canonicity".

To each closed term $t : A$, we associate an element $t' \in A'(t)$.

$$\begin{aligned} \mathbb{N}'(t) &\coloneqq \{n \mid t \equiv \mathsf{suc}^n\, 0\} \\ \big((\Pi x : A).B\big)'(t) &\coloneqq (\Pi a : A)(\Pi a' : A'(a)).B'(a, a')\,(t\ a) \end{aligned}$$

$$\begin{aligned} (t\ a)' &\coloneqq t'\ a\ a' \\ \big((\lambda x : A).t\big)' &\coloneqq (\lambda a : A)(\lambda a' : A'(a)).t'\ a\ a' \\ (\mathsf{suc}\ n)' &\coloneqq n' + 1 \\ &\vdots \end{aligned}$$

By induction on derivations, we can show that if $\vdash a : A$ then $a' \in A'(t)$ and if $\vdash a \equiv b : A$ then $a' = b'$. (Need to generalise statement to closing substitutions.) In particular if $\vdash t : \mathbb{N}$ then $t \equiv \mathsf{suc}^n\, 0$ for some n.

A more structured approach?

We can organise the argument as follows:

For each model $\mathcal{M} = (\mathcal{C}, \mathsf{Ty}, \mathsf{Tm})$, we build a new "canonicity" model $\mathcal{M}^* = (\mathcal{C}^*, \mathsf{Ty}^*, \mathsf{Tm}^*)$ together with a model morphism $\mathcal{M}^* \to \mathcal{M}$.

This way, it is easier to not accidentally forget a clause.

The “canonicity” model

The objects of $\mathcal{C}^*$ are pairs (Γ, Γ') where $\Gamma \in \mathcal{C}$ and $\Gamma' : \mathrm{Hom}_{\mathcal{C}}(1, \Gamma) \to \mathrm{Set}$, with $1^* = (1, \lambda\_.\mathbf{1})$.

Morphisms are pairs (σ, σ') where

$$\begin{aligned} \sigma &: \Delta \to \Gamma \\ \sigma' &: (\Pi\tau : 1 \to \Delta).\big(\Delta'(\tau) \to \Gamma'(\sigma \circ \tau)\big) \end{aligned}$$

The "canonicity" model

The objects of $\mathcal{C}^*$ are pairs (Γ, Γ') where $\Gamma \in \mathcal{C}$ and $\Gamma' : \mathrm{Hom}_{\mathcal{C}}(1, \Gamma) \to \mathrm{Set}$, with $1^* = (1, \lambda\_.\mathbf{1})$.

Morphisms are pairs (σ, σ') where

$$\begin{aligned} \sigma &: \Delta \to \Gamma \\ \sigma' &: (\Pi \tau : 1 \to \Delta).\big(\Delta'(\tau) \to \Gamma'(\sigma \circ \tau)\big) \end{aligned}$$

We define $\mathrm{Ty}^*(\Gamma, \Gamma')$ to be the set of pairs (A, A') where

$$\begin{aligned} A &\in \mathrm{Ty}(\Gamma) \\ A' &\in (\Pi \sigma : 1 \to \Gamma)\big(\Gamma'(\sigma) \to \mathrm{Tm}(1, A[\sigma]) \to \mathrm{Set}\big) \end{aligned}$$

Similarly $\mathrm{Tm}^*((\Gamma, \Gamma'), (A, A'))$ consists of (t, t') such that

$$\begin{aligned} t &\in \mathrm{Tm}(\Gamma, A) \\ t' &\in (\Pi \sigma : 1 \to \Gamma)(\Pi \sigma' \in \Gamma'(\sigma)).A' \, \sigma \, \sigma' \, (t[\sigma]) \end{aligned}$$

Canonicity from $\mathcal{M}^*$

If $\mathcal{M}$ has natural numbers $\mathsf{Nat} \in \mathsf{Ty}(\Gamma)$, we can define $(\mathsf{Nat}, \mathsf{Nat}') \in \mathsf{Ty}^*(\Gamma, \Gamma')$ where

$$\mathsf{Nat}'\,\sigma\,\sigma'\,t :\equiv \{n \mid t \equiv \mathsf{suc}^n\,0\}$$

and similarly for other type and term constructors. The model morphism $\pi : \mathcal{M}^* \to \mathcal{M}$ is given by first projection.

Canonicity from $\mathcal{M}^*$

If $\mathcal{M}$ has natural numbers $\mathsf{Nat} \in \mathsf{Ty}(\Gamma)$, we can define $(\mathsf{Nat}, \mathsf{Nat}') \in \mathsf{Ty}^*(\Gamma, \Gamma')$ where

$$\mathsf{Nat}'\,\sigma\,\sigma'\,t :\equiv \{n \mid t \equiv \mathsf{suc}^n\,0\}$$

and similarly for other type and term constructors. The model morphism $\pi : \mathcal{M}^* \to \mathcal{M}$ is given by first projection.

Theorem In the syntax, every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\,0$.

Canonicity from $\mathcal{M}^*$

If $\mathcal{M}$ has natural numbers $\mathsf{Nat} \in \mathsf{Ty}(\Gamma)$, we can define $(\mathsf{Nat}, \mathsf{Nat}') \in \mathsf{Ty}^*(\Gamma, \Gamma')$ where

$$\mathsf{Nat}'\,\sigma\,\sigma'\,t := \{n \mid t \equiv \mathsf{suc}^n\,0\}$$

and similarly for other type and term constructors. The model morphism $\pi : \mathcal{M}^* \to \mathcal{M}$ is given by first projection.

Theorem In the syntax, every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\,0$.

Proof: The syntax forms an initial model $\mathcal{M}_0$. We thus have a map $i : \mathcal{M}_0 \to \mathcal{M}_0^*$, and $\pi \circ i = \mathsf{id}_{\mathcal{M}_0}$ by initiality. For closed terms $t : \mathbb{N}$ we thus have $t' \in \mathbb{N}' \star \star\, t$ so $t \equiv \mathsf{suc}^n\,0$ for some $n \in \mathbb{N}$. □

Canonicity from $\mathcal{M}^*$

If $\mathcal{M}$ has natural numbers $\mathsf{Nat} \in \mathsf{Ty}(\Gamma)$, we can define $(\mathsf{Nat}, \mathsf{Nat}') \in \mathsf{Ty}^*(\Gamma, \Gamma')$ where

$$\mathsf{Nat}'\,\sigma\,\sigma'\,t :\equiv \{n \mid t \equiv \mathsf{suc}^n\,0\}$$

and similarly for other type and term constructors. The model morphism $\pi : \mathcal{M}^* \to \mathcal{M}$ is given by first projection.

Theorem In the syntax, every closed term of type $\mathbb{N}$ is (judgementally) equal to a numeral $\mathsf{suc}^n\,0$.

Proof: The syntax forms an initial model $\mathcal{M}_0$. We thus have a map $i : \mathcal{M}_0 \to \mathcal{M}_0^*$, and $\pi \circ i = \mathsf{id}_{\mathcal{M}_0}$ by initiality. For closed terms $t : \mathbb{N}$ we thus have $t' \in \mathbb{N}' \star \star\, t$ so $t \equiv \mathsf{suc}^n\,0$ for some $n \in \mathbb{N}$. □

Even more abstractly, this model construction is an instance of gluing for CwFs (Kaposi, Huber, and Sattler 2019) .

Summary

We have seen four models of type theory in the CwF framework:

1. Truth-value model demonstrating the independence of $0 = \mathsf{suc}\, n$ without universes.
2. Groupoid model demonstrating the independence of UIP, and suggesting the "universe extensionality axiom"
3. D-sets model enabling the extraction of computable data, and demonstrating the independence of classical logic.
4. Canonicity model allowing us to derive canonicity.

References

Michael Beeson. "Recursive models for constructive set theories". In: *Annals of Mathematical Logic* 23.2 (1982), pp. 127–178. DOI: 10.1016/0003-4843(82)90003-1.

Marc Bezem, Thierry Coquand, and Simon Huber. "A Model of Type Theory in Cubical Sets". In: *TYPES 2013*. Ed. by Ralph Matthes and Aleksy Schubert. Vol. 26. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013, pp. 107–128. DOI: 10.4230/LIPICS.TYPES.2013.107.

Thierry Coquand. "Pattern matching with dependent types". In: *Informal proceedings of Logical Frameworks*. Vol. 92. 1992, pp. 66–79.

Martin Hofmann and Thomas Streicher. "The Groupoid Model Refutes Uniqueness of Identity Proofs". In: *LICS 1994*. IEEE Computer Society, 1994, pp. 208–212. DOI: 10.1109/LICS.1994.316071.

Ambrus Kaposi, Simon Huber, and Christian Sattler. "Gluing for Type Theory". In: *FSCD 2019*. Ed. by Herman Geuvers. Vol. 131. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 25:1–25:19. DOI: 10.4230/LIPIcs.FSCD.2019.25.

Krzysztof Kapulkin and Peter LeFanu Lumsdaine. "The simplicial model of Univalent Foundations (after Voevodsky)". In: *Journal of the European Mathematical Society* 23.6 (2021), pp. 2071–2126. DOI: 10.4171/JEMS/1050.

Conor McBride. "Dependently Typed Functional Programs and Their Proofs". PhD thesis. University of Edinburgh, 1999.

Jan M. Smith. "The Independence of Peano's Fourth Axiom from Martin-Löf's Type Theory Without Universes". In: *The Journal of Symbolic Logic* 53.3 (1988), pp. 840–845.

Thomas Streicher. *Investigations into intensional type theory*. Habilitation thesis. 1993.