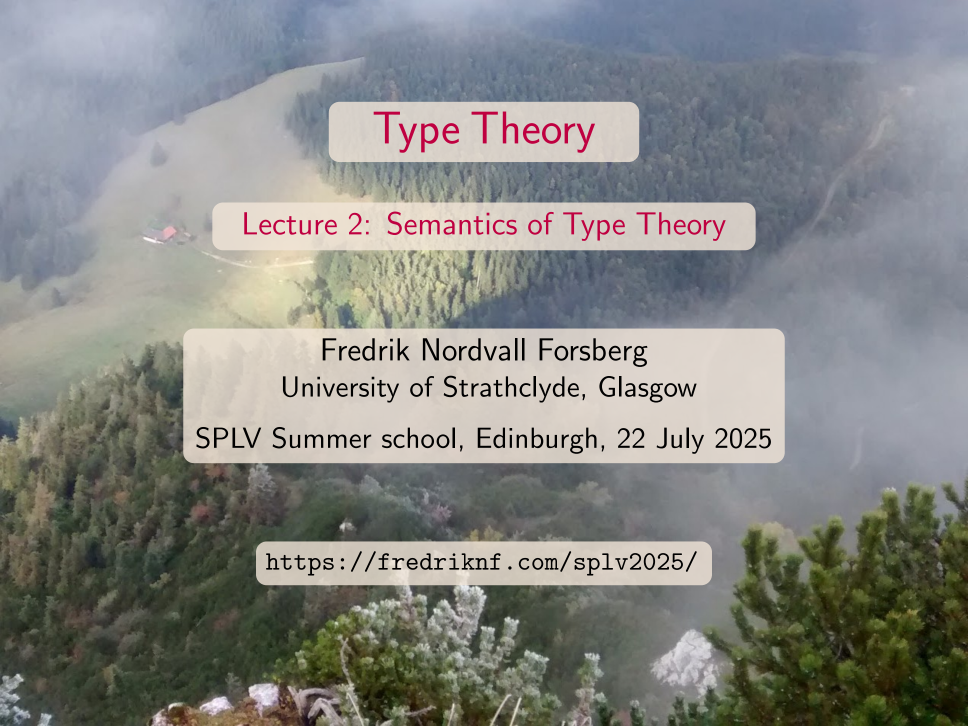

Type Theory

Lecture 2: Semantics of Type Theory

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

SPLV Summer school, Edinburgh, 22 July 2025

https://fredriknf.com/splv2025/

Course plan

- **Yesterday:** Using type theory.
- **Today:** Semantics of type theory.
  - Categorical framework for models
  - Some concrete models, and what they are good for
- **Thursday:** Implementation and metatheory.

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

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

Set-theoretic model:

- For each type A, define set $[\![A]\!]$ (canonical def. for $\to$ and $\times$);

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

Set-theoretic model:

- For each type A, define set $[\![A]\!]$ (canonical def. for $\to$ and $\times$);
- for each context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, define $[\![\Gamma]\!] = [\![A_1]\!] \times \ldots [\![A_n]\!]$; and

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

Set-theoretic model:

- For each type A, define set $[\![A]\!]$ (canonical def. for $\to$ and $\times$);
- for each context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, define $[\![\Gamma]\!] = [\![A_1]\!] \times \ldots [\![A_n]\!]$; and
- for each term $\Gamma \vdash t : A$, define a function $[\![\Gamma]\!] \to [\![A]\!]$.

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

Set-theoretic model:

- For each type A, define set $[\![A]\!]$ (canonical def. for $\to$ and $\times$);
- for each context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, define $[\![\Gamma]\!] = [\![A_1]\!] \times \ldots [\![A_n]\!]$; and
- for each term $\Gamma \vdash t : A$, define a function $[\![\Gamma]\!] \to [\![A]\!]$.
- **Soundness:** If $\Gamma \vdash t = u : A$, then $[\![t]\!] = [\![u]\!]$.
- **Completeness:** If $[\![t]\!]_{\mathcal{M}} = [\![u]\!]_{\mathcal{M}}$ for all models $\mathcal{M}$, do we have $\Gamma \vdash t = u : A$?

What is a model of simple type theory?

Simple types: function types $A \to B$, product types $A \times B$ (maybe a base type, say ι).

Set-theoretic model:

- For each type A, define set $[\![A]\!]$ (canonical def. for $\to$ and $\times$);
- for each context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, define $[\![\Gamma]\!] = [\![A_1]\!] \times \ldots [\![A_n]\!]$; and
- for each term $\Gamma \vdash t : A$, define a function $[\![\Gamma]\!] \to [\![A]\!]$.
- **Soundness:** If $\Gamma \vdash t = u : A$, then $[\![t]\!] = [\![u]\!]$.
- **Completeness:** If $[\![t]\!]_{\mathcal{M}} = [\![u]\!]_{\mathcal{M}}$ for all models $\mathcal{M}$, do we have $\Gamma \vdash t = u : A$?

Completeness in this form is true [Friedman 1975], but quite hard to prove (since we need to use the full function space).

Models of simple types in Cartesian Closed Categories

Rather than insisting on interpreting types as sets, we can broaden our notion of model.

This makes Completeness weaker (and easier to prove), but Soundness stronger.

Models of simple types in Cartesian Closed Categories

Rather than insisting on interpreting types as sets, we can broaden our notion of model.

This makes Completeness weaker (and easier to prove), but Soundness stronger.

Cartesian closure: A category $\mathcal{C}$ is Cartesian closed if it has

- A terminal object $\mathbf{1}$
- Binary products $A \times B$
- Exponentials $A \Rightarrow B$

Models of simple types in Cartesian Closed Categories

Rather than insisting on interpreting types as sets, we can broaden our notion of model.

This makes Completeness weaker (and easier to prove), but Soundness stronger.

Cartesian closure: A category $\mathcal{C}$ is Cartesian closed if it has

- A terminal object $\mathbf{1}$
- Binary products $A \times B$
- Exponentials $A \Rightarrow B$

Exactly what we need to interpret the simply typed λ-calculus!

Models of simple types in Cartesian Closed Categories

Rather than insisting on interpreting types as sets, we can broaden our notion of model.

This makes Completeness weaker (and easier to prove), but Soundness stronger.

Cartesian closure: A category $\mathcal{C}$ is Cartesian closed if it has

- A terminal object $\mathbf{1}$
- Binary products $A \times B$
- Exponentials $A \Rightarrow B$

Exactly what we need to interpret the simply typed λ-calculus!

Soundness and completeness: $\Gamma \vdash t = u : A$ iff $[\![t]\!]_{\mathcal{C}} = [\![u]\!]_{\mathcal{C}}$ for every Cartesian closed category $\mathcal{C}$.

What is a model of dependent type theory?

What is a model of dependent type theory?

As usual, things are more intricate for dependent types.

Categories with families were introduced by Peter Dybjer [1995].

Inspired by *contextual categories*, *categories with attributes* and *generalised algebraic theories* by John Cartmell [1978].

Main idea: What is fundamental is the category of contexts.

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.

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.

Together, Ty and Tm constitute a functor

$$(\mathsf{Ty}, \mathsf{Tm}) : \mathcal{C}^{\mathsf{op}} \to \mathsf{Fam}\,\mathsf{Set}$$

to the category of families of sets, hence the name.

Unpacking the definition: the category $\mathcal{C}$

Intuition:

- **Objects** (Interpretation of) contexts
- **Morphisms** (Interpretation of) substitutions

In the syntax, a substitution $\Gamma \to \Delta$ with $\Delta = x_1 : A_1, \ldots, x_n : A_n$ is given by a sequence of terms $(t_1, \ldots, t_n)$ with

$$\begin{gathered} \Gamma \vdash t_1 : A_1 \\ \Gamma \vdash t_2 : A_2[x_1 \mapsto t_1] \\ \vdots \end{gathered}$$

In particular, there is a unique substitution $\Gamma \to 1$ to the empty context 1 for every Γ — 1 is a terminal object.

Unpacking the definition: types

The presheaf $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ gives:

- A set of (semantic) types $\mathsf{Ty}(\Gamma)$ for each (semantic) context $\Gamma \in \mathcal{C}$.
- For each $\sigma : \Delta \to \Gamma$, a function $\_[\sigma] : \mathsf{Ty}(\Gamma) \to \mathsf{Ty}(\Delta)$,
- such that $A[\mathsf{id}] = A$ and $A[\sigma][\tau] = A[\sigma \circ \tau]$.

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$.

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$. Morphisms are underlying morphisms preserving the element.

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$. Morphisms are underlying morphisms preserving the element.

Hence, the presheaf $\mathsf{Tm} : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$ gives:

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{op} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$. Morphisms are underlying morphisms preserving the element.

Hence, the presheaf $\mathsf{Tm} : (\int_{\mathcal{C}} \mathsf{Ty})^{op} \to \mathsf{Set}$ gives:

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, a set $\mathsf{Tm}(\Gamma, A)$.

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$. Morphisms are underlying morphisms preserving the element.

Hence, the presheaf $\mathsf{Tm} : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$ gives:

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, a set $\mathsf{Tm}(\Gamma, A)$.
- For each $\sigma : \Delta \to \Gamma$, a function $\_[\sigma] : \mathsf{Tm}(\Gamma, A) \to \mathsf{Tm}(\Delta, A[\sigma])$,

Unpacking the definition: terms

Definition Given a functor $F : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, the category of elements $\int_{\mathcal{C}} F$ has as objects pairs (Γ, A) where $\Gamma \in \mathcal{C}$ and $A \in F(\Gamma)$. Morphisms are underlying morphisms preserving the element.

Hence, the presheaf $\mathsf{Tm} : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$ gives:

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, a set $\mathsf{Tm}(\Gamma, A)$.
- For each $\sigma : \Delta \to \Gamma$, a function $\_[\sigma] : \mathsf{Tm}(\Gamma, A) \to \mathsf{Tm}(\Delta, A[\sigma])$,
- such that $t[\mathrm{id}] = t$ and $t[\sigma][\tau] = t[\sigma \circ \tau]$.

(These equations make sense because of the equations for types.)

Context extension

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, we have an object $\Gamma \cdot A \in \mathcal{C}$.

Context extension

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, we have an object $\Gamma \cdot A \in \mathcal{C}$.
- Further, there is a "projection" $\mathsf{p}_{\Gamma,A} : \Gamma \cdot A \to \Gamma$ in $\mathcal{C}$,

Context extension

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, we have an object $\Gamma \cdot A \in \mathcal{C}$.
- Further, there is a "projection" $\mathsf{p}_{\Gamma,A} : \Gamma \cdot A \to \Gamma$ in $\mathcal{C}$,
- and a term $\mathsf{q}_{\Gamma,A} \in \mathsf{Tm}(\Gamma \cdot A, A[\mathsf{p}_{\Gamma,A}])$,

Context extension

- For each $\Gamma \in \mathcal{C}$ and $A \in \mathsf{Ty}(\Gamma)$, we have an object $\Gamma \cdot A \in \mathcal{C}$.
- Further, there is a "projection" $\mathsf{p}_{\Gamma,A} : \Gamma \cdot A \to \Gamma$ in $\mathcal{C}$,
- and a term $\mathsf{q}_{\Gamma,A} \in \mathsf{Tm}(\Gamma \cdot A, A[\mathsf{p}_{\Gamma,A}])$,
- and if $\sigma : \Delta \to \Gamma$ and $u \in \mathsf{Tm}(\Delta, A[\sigma])$ then there is a unique morphism $\langle \sigma, u \rangle : \Delta \to \Gamma \cdot A$ such that $\mathsf{p} \circ \langle \sigma, u \rangle = \sigma$ and $\mathsf{q}[\langle \sigma, u \rangle] = u$.

Some useful constructions

Given $t \in \mathsf{Tm}(\Gamma, A)$, we can construct $\overline{t} \coloneqq \langle \mathsf{id}, t \rangle : \Gamma \to \Gamma \cdot A$ which "plugs in t": if $B \in \mathsf{Ty}(\Gamma \cdot A)$ then $B[\,\overline{t}\,] \in \mathsf{Ty}(\Gamma)$.

Some useful constructions

Given $t \in \mathsf{Tm}(\Gamma, A)$, we can construct $\overline{t} := \langle \mathsf{id}, t \rangle : \Gamma \to \Gamma \cdot A$ which "plugs in t": if $B \in \mathsf{Ty}(\Gamma \cdot A)$ then $B[\,\overline{t}\,] \in \mathsf{Ty}(\Gamma)$.

Given $\sigma : \Delta \to \Gamma$ and $A \in \mathsf{Ty}(\Gamma)$, we can construct $\sigma^+ := \langle \sigma \circ \mathsf{p}, \mathsf{q} \rangle : \Delta \cdot A[\sigma] \to \Gamma \cdot A$ which "lifts σ under binders".

Some useful constructions

Given $t \in \mathsf{Tm}(\Gamma, A)$, we can construct $\overline{t} := \langle \mathsf{id}, t \rangle : \Gamma \to \Gamma \cdot A$ which "plugs in t": if $B \in \mathsf{Ty}(\Gamma \cdot A)$ then $B[\,\overline{t}\,] \in \mathsf{Ty}(\Gamma)$.

Given $\sigma : \Delta \to \Gamma$ and $A \in \mathsf{Ty}(\Gamma)$, we can construct $\sigma^{+} := \langle \sigma \circ \mathsf{p}, \mathsf{q} \rangle : \Delta \cdot A[\sigma] \to \Gamma \cdot A$ which "lifts σ under binders".

Exercise

The following diagram commutes, and is in fact a pullback:

$$\begin{array}{ccc}
\Delta \cdot A[\sigma] & \xrightarrow{\sigma^{+}} & \Gamma \cdot A \\
{\scriptstyle \mathsf{p}} \downarrow & & \downarrow {\scriptstyle \mathsf{p}} \\
\Delta & \xrightarrow[\sigma]{} & \Gamma
\end{array}$$

The Set model

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

The Set model

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

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

The Set model

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

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

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

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

The Set model

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

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

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

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

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

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 $\mathsf{p} :\equiv \mathsf{fst}$, $\mathsf{q} :\equiv \mathsf{snd}$.

Additional type structure

A "pure" CwF does not actually interpret any type formers; we have to ask for those on top.

But we now have the language needed to translate syntactic to semantic notions.

Additional type structure

A "pure" CwF does not actually interpret any type formers; we have to ask for those on top.

But we now have the language needed to translate syntactic to semantic notions.

(Often there is also a more elegant equivalent "semantic" criterion, see e.g. Awodey's work on so-called natural models (2018).)

Dependent function types

Definition A CwF $\mathcal{C}$ supports dependent function types if

- for all $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma \cdot A)$ there is $\Pi\, A\, B \in \mathsf{Ty}(\Gamma)$,

Dependent function types

Definition A CwF $\mathcal{C}$ supports dependent function types if

- for all $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma \cdot A)$ there is $\Pi\, A\, B \in \mathsf{Ty}(\Gamma)$,
- for all $t \in \mathsf{Tm}(\Gamma \cdot A, B)$ there is $\lambda_{A,B}(t) \in \mathsf{Tm}(\Gamma, \Pi\, A\, B)$,

Dependent function types

Definition A CwF $\mathcal{C}$ supports dependent function types if

- for all $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma \cdot A)$ there is $\Pi\, A\, B \in \mathsf{Ty}(\Gamma)$,
- for all $t \in \mathsf{Tm}(\Gamma \cdot A, B)$ there is $\lambda_{A,B}(t) \in \mathsf{Tm}(\Gamma, \Pi\, A\, B)$,
- for all $f \in \mathsf{Tm}(\Gamma, \Pi\, A\, B)$ and $u \in \mathsf{Tm}(\Gamma, A)$, there is $\mathsf{App}_{A,B}(f, u) \in \mathsf{Tm}(\Gamma, B[\overline{u}])$

Dependent function types

Definition A CwF $\mathcal{C}$ supports dependent function types if

- for all $A \in \mathsf{Ty}(\Gamma)$ and $B \in \mathsf{Ty}(\Gamma \cdot A)$ there is $\Pi\, A\, B \in \mathsf{Ty}(\Gamma)$,
- for all $t \in \mathsf{Tm}(\Gamma \cdot A, B)$ there is $\lambda_{A,B}(t) \in \mathsf{Tm}(\Gamma, \Pi\, A\, B)$,
- for all $f \in \mathsf{Tm}(\Gamma, \Pi\, A\, B)$ and $u \in \mathsf{Tm}(\Gamma, A)$, there is $\mathsf{App}_{A,B}(f, u) \in \mathsf{Tm}(\Gamma, B[\overline{u}])$
- such that

$$
\begin{aligned}
(\Pi\, A\, B)[\sigma] &= \Pi\,(A[\sigma])\,(B[\sigma^+]) \\
(\lambda_{A,B}(t))[\sigma] &= \lambda_{A[\sigma],B[\sigma^+]}(t[\sigma^+]) \\
(\mathsf{App}_{A,B}(f, u))[\sigma] &= \mathsf{App}_{A[\sigma],B[\sigma^+]}(f[\sigma], u[\sigma^+]) \\
\mathsf{App}_{A,B}(\lambda_{A,B}(t), u) &= t[\overline{u}] \\
\lambda_{A,B}(\mathsf{App}_{A,B}(t[\mathsf{p}], \mathsf{q})) &= t
\end{aligned}
$$

The empty type

Definition A CwF $\mathcal{C}$ supports the empty type if

- there is $\mathsf{Empty} \in \mathsf{Ty}(\Gamma)$,

The empty type

Definition A CwF $\mathcal{C}$ supports the empty type if

- there is $\mathsf{Empty} \in \mathsf{Ty}(\Gamma)$,
- for all $C \in \mathsf{Ty}(\Gamma)$ and $p \in \mathsf{Tm}(\Gamma, \mathsf{Empty})$ there is $\mathsf{elim}_{\mathsf{Empty}}(C, p) \in \mathsf{Tm}(\Gamma, C)$,

The empty type

Definition A CwF $\mathcal{C}$ supports the empty type if

- there is $\mathsf{Empty} \in \mathsf{Ty}(\Gamma)$,
- for all $C \in \mathsf{Ty}(\Gamma)$ and $p \in \mathsf{Tm}(\Gamma, \mathsf{Empty})$ there is $\mathsf{elim}_{\mathsf{Empty}}(C, p) \in \mathsf{Tm}(\Gamma, C)$,
- such that

$$\mathsf{Empty}[\sigma] = \mathsf{Empty}$$
$$(\mathsf{elim}_{\mathsf{Empty}}(C, p))[\sigma] = \mathsf{elim}_{\mathsf{Empty}}(C[\sigma], p[\sigma])$$

The empty type

Definition A CwF $\mathcal{C}$ supports the empty type if

- there is $\mathsf{Empty} \in \mathsf{Ty}(\Gamma)$,
- for all $C \in \mathsf{Ty}(\Gamma)$ and $p \in \mathsf{Tm}(\Gamma, \mathsf{Empty})$ there is $\mathsf{elim}_{\mathsf{Empty}}(C, p) \in \mathsf{Tm}(\Gamma, C)$,
- such that

$$\begin{aligned} \mathsf{Empty}[\sigma] &= \mathsf{Empty} \\ (\mathsf{elim}_{\mathsf{Empty}}(C, p))[\sigma] &= \mathsf{elim}_{\mathsf{Empty}}(C[\sigma], p[\sigma]) \end{aligned}$$

and similarly for the natural numbers, etc.

The identity type

Definition A CwF $\mathcal{C}$ supports identity types if for every $A \in \mathsf{Ty}(\Gamma)$,

- there is $\mathsf{Id}_A \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}])$,

The identity type

Definition A CwF $\mathcal{C}$ supports identity types if for every $A \in \mathsf{Ty}(\Gamma)$,

- there is $\mathsf{Id}_A \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}])$,
- and $\mathsf{refl} \in \mathsf{Tm}(\Gamma \cdot A, \mathsf{Id}_A[\langle \mathsf{id}_{\Gamma . A}, \mathsf{q} \rangle])$,

The identity type

Definition A CwF $\mathcal{C}$ supports identity types if for every $A \in \mathsf{Ty}(\Gamma)$,

- there is $\mathsf{Id}_A \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}])$,
- and $\mathsf{refl} \in \mathsf{Tm}(\Gamma \cdot A, \mathsf{Id}_A[\langle \mathsf{id}_{\Gamma \cdot A}, \mathsf{q} \rangle])$,
- and for each $C \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}] \cdot \mathsf{Id}_A)$, there is $\mathsf{elim}_= : \mathsf{Tm}(\Gamma \cdot A, C[\langle \langle \mathsf{id}, \mathsf{q} \rangle, \mathsf{refl} \rangle]) \to \mathsf{Tm}(\Gamma \cdot A \cdot A[\mathsf{p}] \cdot \mathsf{Id}_A, C)$

The identity type

Definition A CwF $\mathcal{C}$ supports identity types if for every $A \in \mathsf{Ty}(\Gamma)$,

- there is $\mathsf{Id}_A \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}])$,
- and $\mathsf{refl} \in \mathsf{Tm}(\Gamma \cdot A, \mathsf{Id}_A[\langle \mathsf{id}_{\Gamma \cdot A}, \mathsf{q}\rangle])$,
- and for each $C \in \mathsf{Ty}(\Gamma \cdot A \cdot A[\mathsf{p}] \cdot \mathsf{Id}_A)$, there is $\mathsf{elim}_= : \mathsf{Tm}(\Gamma \cdot A, C[\langle\langle \mathsf{id}, \mathsf{q}\rangle, \mathsf{refl}\rangle]) \to \mathsf{Tm}(\Gamma \cdot A \cdot A[\mathsf{p}] \cdot \mathsf{Id}_A, C)$
- all stable under substitution.

The Set model again

The **Set** model supports all type formers we have considered as follows:

The Set model again

The **Set** model supports all type formers we have considered as follows:

Given $A : \Gamma \to \mathsf{Set}$ and $B : (\Sigma\gamma \in \Gamma).A(\gamma) \to \mathsf{Set}$, define

$$(\Pi\, A\, B)\, \gamma :\equiv (\Pi x \in A(\gamma)).B(\gamma, x)$$

The Set model again

The **Set** model supports all type formers we have considered as follows:

Given $A : \Gamma \to \text{Set}$ and $B : (\Sigma\gamma \in \Gamma).A(\gamma) \to \text{Set}$, define

$$(\Pi\, A\, B)\, \gamma \coloneqq (\Pi x \in A(\gamma)).B(\gamma, x)$$

Given $A : \Gamma \to \text{Set}$, and $a, b \in (\Pi\gamma \in \Gamma).A(\gamma)$, define

$$\mathsf{Id}(A, a, b)\, \gamma = \{\star \mid a(\gamma) = b(\gamma)\}$$

The Set model again

The **Set** model supports all type formers we have considered as follows:

Given $A : \Gamma \to \mathsf{Set}$ and $B : (\Sigma\gamma \in \Gamma).A(\gamma) \to \mathsf{Set}$, define

$$(\Pi\, A\, B)\, \gamma :\equiv (\Pi x \in A(\gamma)).B(\gamma, x)$$

Given $A : \Gamma \to \mathsf{Set}$, and $a, b \in (\Pi\gamma \in \Gamma).A(\gamma)$, define

$$\mathsf{Id}(A, a, b)\, \gamma = \{\star \mid a(\gamma) = b(\gamma)\}$$

The empty type, natural numbers can be interpreted by defining $\mathsf{Empty} : \Gamma \to \mathsf{Set}$, $\mathsf{Nat} : \Gamma \to \mathsf{Set}$ by

$$\mathsf{Empty}\, \gamma :\equiv \emptyset \qquad\qquad \mathsf{Nat}\, \gamma :\equiv \mathbb{N}$$

Constructions on models

The notion of CwF (plus type structure) is a generalised algebraic theory (Cartmell 1978), thus very well behaved:

There is a canonical notion of morphism of models (preserving all the structure).

We can take the product of models.

There is an initial model: the syntax.

Constructions on models

The notion of CwF (plus type structure) is a generalised algebraic theory (Cartmell 1978), thus very well behaved:

There is a canonical notion of morphism of models (preserving all the structure).

We can take the product of models.

There is an initial model: the syntax.

Theorem (soundness and completeness) A judgement holds in the syntax iff it holds in all models.

Completeness is practically useless, but something would be wrong if we did not have it.

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

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

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

We define

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

and

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

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

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

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! $\leadsto$ ∞-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 $\mathcal{C} := \mathbf{Gpd}$, the category of groupoids and functors.

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

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

The groupoid model

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

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

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

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

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

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

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 $\Gamma \cdot A := \int_\Gamma A$, 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 $\mathsf{Hom}_A(a, b)$. 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 $\mathsf{Hom}_A(a,b)$. 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 $\mathsf{elim}_=(d,r) \in C(x,y,r)$.

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

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

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 refl : $\mathsf{Id}\,A\,a\,a$, we can take 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

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

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 $U(\gamma) := \mathrm{Gpd}_V$, 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.

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.,

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

We would then have

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

in the model, but $\mathrm{Id}\,B\mathbb{Z}\ \star\ \star$ is a discrete groupoid, hence $\mathrm{Id}\,(\mathrm{Id}\,B\mathbb{Z}\ \star\ \star)\,0\,1 = \emptyset$ 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 $\mathrm{Id}\,A\,a\,b$ 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 $f \$ a_1 \$ \ldots \$ a_n = t(a_1, \ldots, a_n)$.

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 $d \mathbin{\$} a \Vdash_Y f(x)$.

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 $d \mathbin{\$} a \Vdash_Y f(x)$.

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

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

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.

Summary

Categories with families as a framework for models of dependent type theory. (There are many other similar notions.)

Looked at three models:

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.

Thursday: Some implementation, some metatheory.

References

Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2018), pp. 241–286. DOI: 10.1017/S0960129516000268.

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.

John Cartmell. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Oxford University, 1978.

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

Peter Dybjer. "Internal Type Theory". In: *TYPES 1995*. Ed. by Stefano Berardi and Mario Coppo. Vol. 1158. Lecture Notes in Computer Science. Springer, 1995, pp. 120–134. DOI: 10.1007/3-540-61780-9_66.

Harvey M. Friedman. "Equality between functionals". In: *Logic Colloquium '73*. Ed. by R. Parikh. 1975.

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.

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.