Categories with families

Jordan Mitchell Barrett

Originally published: October 14, 2021
This version: October 14, 2021

Contents

1 Introduction

This report aims to provide a gentle introduction to **semantics of type theory** and **categories with families** (CwFs). I've found that existing presentations of this material tend to be terse and mathematically minimal. Of course, this is an efficient way to present that information to experts in the field. However, it assumes context and background which beginners may not have. Here, we aim to bridge that gap.

We will assume the reader has previous exposure to:

- Basic type theory, at the level of Chapter 1 of the HoTT book [6].
- Category theory, as covered in the first few chapters of your favourite textbook [4, 5].

We'll start by talking about the initial (and possibly canonical) model of type theory, the *syntactic model*, in §2. This will serve as the primary motivation for the definition of CwFs in §3. Then, we'll cover two more important models in detail: the *set-theoretical model* in §4, and Hofmann & Streicher's *groupoid model* [3] in §5.

By the end, we hope the reader will be able to make sense of other presentations of CwFs in the literature. Let's dive in!

2 The syntax of a type theory

A *type theory* is a collection of rules for forming *contexts*, *types*, and *terms*. The exact definition of contexts, types and terms depends on the type theory in question. However, the idea is:

- Types are like *sets*, and represent the possible kinds of data we can have. For example, $\mathbb{N}$, $\mathbb{R}$, `Bool`, `String` are types.
- Terms are like *elements*, and represent individual mathematical objects or pieces of data. Terms are allowed to have variables. For example, 3, $3+n$, n^2+4 are all terms (of type $\mathbb{N}$), while $\pi + e + x$ is a term of type $\mathbb{R}$.

Generally, each term t is associated with a unique type A, and we write $t : A$ to say t has type A. We formalise this idea with the notion of a typing judgment.

Definition 2.1. A *typing judgment* is a string $t : A$, where t is a term and A is a type.

2.1 Contexts

Definition 2.2. A *context* Γ is a sequence of variables and associated typing judgments, i.e.

$$\Gamma = \big(x_1 : A_1,\ x_2 : A_2,\ \ldots,\ x_n : A_n\big)$$

In a context Γ, the x_i are only allowed to be variables, not arbitrary terms. Furthermore, each type A_i is allowed to *depend* on the previous variables x_i, so we might write Γ more accurately as

$$\begin{aligned}\Gamma = (\ & x_1 : A_1, \\ & x_2 : A_2(x_1), \\ & x_3 : A_3(x_1, x_2), \\ & \quad \vdots \\ & x_n : A_n(x_1, \ldots, x_{n-1})\ )\end{aligned}$$

We also have a notion of a typing judgment $t : A$ being *valid* in a context Γ, which we write $\Gamma \vdash t : A$. The exact definition of validity is prescribed by the type theory in question, but the idea is that Γ gives us enough information to conclude that t has type A.

Remark 2.3. Not every list of typing judgments is a well-formed context—the rules of the type theory tell us which lists are. Generally, the empty context $()$ is well-formed, and given a context Γ, we can instantiate a new variable x and extend it to $(\Gamma,\, x : A)$, where A is a *valid type*[1] in context Γ.

2.2 Substitutions

Now, we have a notion of substitution, which takes us from one context to another.

Definition 2.4. Given two contexts

$$\begin{aligned}\Gamma = (\ & x_1 : A_1, \\ & x_2 : A_2(x_1), \\ & x_3 : A_3(x_1, x_2), \\ & \quad \vdots \\ & x_n : A_n(x_1, \ldots, x_{n-1})\ )\end{aligned} \qquad \begin{aligned}\Delta = (\ & y_1 : B_1, \\ & y_2 : B_2(y_1), \\ & y_3 : B_3(y_1, y_2), \\ & \quad \vdots \\ & y_m : B_m(y_1, \ldots, y_{m-1})\ )\end{aligned}$$

a *substitution* σ from Γ to Δ is a sequence of terms $(s_1, \ldots, s_m)$ such that

$$\Gamma \vdash s_i : B_i(s_1, \ldots, s_{i-1})$$

for each $i \leq m$.

Essentially, we are substituting each y_i for s_i, hence the name. The intention is that each s_i is a term constructed from variables x_i in Γ. Here are some simple observations about substitutions:

[1] We will discuss what this means in §2.3.

- We can compose substitutions: if $\sigma = (s_i)$ is a substitution from Γ to Δ, and $\tau = (t_j)$ is a substitution from Δ to $\Theta = (z_j : C_j)_{j \leq k}$, then there is a composite substitution $\tau\sigma$ from Γ to Θ, defined by

$$\tau\sigma = \big(t_j[y_i \coloneqq s_i]\big)$$

  i.e. every occurrence of y_i in t_j is substituted by s_i.

- For every context Γ, there is a trivial substitution from Γ to Γ, defined $\mathrm{id}_\Gamma = (x_1, \ldots, x_n)$. Furthermore, id_Γ is a two-sided identity for composition.

- It is laborious (though routine) to prove that composition is associative.

The astute reader will have noticed from the above observations that we have just formed a *category* **Ctx**, with contexts as objects and substitutions as morphisms. This category is known as the *category of contexts* or *syntactic category* of the type theory.

A final observation to make is that for every context Γ, there is a unique substitution from Γ to the empty context $()$, consisting of an empty list of terms. In categorical terms, $()$ is a *terminal object* in our category.

2.3 Types and terms

For every context Γ, we have a collection $\mathrm{Ty}(\Gamma)$ of *valid types in* Γ. The notations "$\Gamma \vdash A$", "$\Gamma \vdash A$ type", or "$A \in \mathrm{Ty}(\Gamma)$" may be used to express that A is a valid type in the context Γ. As usual, the type theory prescribes rules to determine which types are valid in a given context, but the idea is that A can be formed from the types already in Γ.

Each type $A \in \mathrm{Ty}(\Gamma)$ also has an associated set of *terms*

$$\mathrm{Tm}(\Gamma, A) \coloneqq \{t \text{ term} \mid \Gamma \vdash t : A\}$$

A substitution $\sigma = (s_i)$ from $\Gamma = (x_h : A_h)$ to $\Delta = (y_i : B_i)$ induces a function from $\mathrm{Ty}(\Delta)$ to $\mathrm{Ty}(\Gamma)$, as follows. Given a valid type $A \in \mathrm{Ty}(\Delta)$, replace every occurrence of y_i in A with s_i to get $A\sigma$. Furthermore, σ induces a function from A to $A\sigma$ in exactly the same way.

Remark 2.5. Some observations:

- If $\sigma = \mathrm{id}$, then the function on types (or terms) induced by σ is simply the identity function.

- The function on types (or terms) induced by $\delta\sigma$ is the composition[2] of that induced by σ and that induced by δ.

You might already notice that this has the structure of a functor. Indeed, we can package it all up nicely in the following way:

[2]Here, we mean regular function composition in **Set**.

Definition 2.6. Fam is "the category of families of sets", where:

- The objects are families of sets $(A_i)_{i \in I}$ indexed by a set I.
- The morphisms from $(A_i)_{i \in I}$ to $(B_j)_{j \in J}$ consist of a function $f \colon I \to J$, and for each i, a function $g_i \colon A_i \to B_{f(i)}$.

Definition 2.7. Define a map $F \colon \mathbf{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$ as follows:

- For a context Γ, $F(\Gamma)$ consists of the sets $\mathrm{Tm}(\Gamma, A)$ indexed by each $A \in \mathrm{Ty}(\Gamma)$.
- For a substitution σ from Γ to Δ, $F(\sigma)$ is the map outlined in Remark 2.5.

Proposition 2.8. F is a functor $\mathbf{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$, called the *term functor* $\mathrm{Tm} \colon \mathbf{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$.

2.4 Context extension and projection

Given any context Γ and valid type A in Γ, we can define an *extended context* $(\Gamma,\, x : A)$. This is also sometimes denoted as $\Gamma.A$.

- Then, we have a substitution π from $(\Gamma,\, x : A)$ to Γ, defined by substituting each x_i with x_i (and ignoring x). We call π the *(first) projection*.
- Furthermore, we get a term $q \in \mathrm{Tm}\big((\Gamma,\, x : A), A\pi\big)$ by $q \coloneqq x$. We sometimes call q the *second projection*.
- Now, given a substitution σ from Δ to Γ, and a term $t \in \mathrm{Tm}(\Delta, A\sigma)$, we can extend σ to a substitution $\langle \sigma, t \rangle$ from Δ to $(\Gamma,\, x : A)$, where x is substituted by t.
- What happens if we compose $\langle \sigma, t \rangle$ and π? Every x_i is replaced by the ith term of $\langle \sigma, t \rangle$, which is just the ith term of σ. Hence,

$$\pi \langle \sigma, t \rangle = \sigma \tag{1}$$

- What about if we apply $\langle \sigma, t \rangle$ to q? Remember, $q = x$, and the sole occurrence of x is replaced with t, so

$$q \langle \sigma, t \rangle = t \tag{2}$$

Try to convince yourself that:

- $\langle \sigma, t \rangle$ is the unique substitution satisfying equations (1) and (2).
- $\langle \pi, q \rangle = \mathrm{id}_{(\Gamma,\, x:A)}$.
- $\langle \sigma, t \rangle \tau = \big\langle \sigma\tau, t\tau \big\rangle$.

We can express all this in a more categorical way. We've essentially defined an isomorphism between the sets

$$\begin{aligned} M(\Delta, \Gamma, A) &\coloneqq \left\{(\sigma, t) \;\middle|\; \sigma \text{ subst. } \Delta \to \Gamma,\ t \in \mathrm{Tm}(\Delta, A\sigma)\right\} \\ \mathrm{Hom}\left(\Delta, (\Gamma,\ x : A)\right) &\coloneqq \{\text{substitutions from } \Delta \text{ to } (\Gamma,\ x : A)\} \end{aligned}$$

To go from $M(\Delta, \Gamma, A)$ to $\mathrm{Hom}\left(\Delta, (\Gamma,\ x : A)\right)$, we use the extended substitution as defined above, while in the other direction, we compose with π and q.

Both of these can be considered as functors $\mathbf{Ctx}^{\mathrm{op}} \to \mathbf{Set}$

$$\begin{aligned} M(-, \Gamma, A) \colon\ \Delta &\mapsto M(\Delta, \Gamma, A) \\ \mathrm{Hom}\left(-, (\Gamma,\ x : A)\right) \colon\ \Delta &\mapsto \mathrm{Hom}\left(\Delta, (\Gamma,\ x : A)\right) \end{aligned}$$

with substitutions acting by composition. Now, the isomorphism is a *natural isomorphism* between $M(-, \Gamma, A)$ and $\mathrm{Hom}\left(-, (\Gamma,\ x\ :\ A)\right)$ [3], i.e. for every substitution τ from Δ to Δ', the following diagram commutes:

$$\begin{array}{ccc} \Delta & M(\Delta, \Gamma, A) & \xrightarrow{\langle\cdot,\cdot\rangle} \mathrm{Hom}\left(\Delta, (\Gamma,\ x : A)\right) \\ \Big\downarrow{\scriptstyle \tau} & \Big\uparrow{\scriptstyle (\cdot)\tau} & \Big\uparrow{\scriptstyle (\cdot)\tau} \\ \Delta' & M(\Delta', \Gamma, A) & \xrightarrow{\langle\cdot,\cdot\rangle} \mathrm{Hom}\left(\Delta', (\Gamma,\ x : A)\right) \end{array}$$

Another nice categorical way to package this up is as follows. Recall that for $\Gamma \in \mathbf{Ctx}$, the *slice category* $\mathbf{Ctx}/\Gamma$ is the category whose:

- Objects are substitutions $\sigma \colon \Delta \to \Gamma$, for some $\Delta \in \mathbf{Ctx}$;
- Morphisms from $\sigma \colon \Delta \to \Gamma$ to $\sigma' \colon \Delta' \to \Gamma$ are substitutions $\tau \colon \Delta \to \Delta'$ such that $\sigma'\tau = \sigma$.

Now, for every $A \in \mathrm{Ty}(\Gamma)$, we can see $M(-, \Gamma, A)$ as a functor $(\mathbf{Ctx}/\Gamma)^{\mathrm{op}} \to \mathbf{Set}$, which sends $\sigma \colon \Delta \to \Gamma$ to $\mathrm{Tm}(\Delta, A\sigma)$, and sends morphisms $\tau \colon \Delta \to \Delta'$ to the map they induce on terms (§2.3). Above, we saw this functor M was naturally isomorphic to the *hom-set functor* $\mathrm{Hom}(-, \pi) \colon (\mathbf{Ctx}/\Gamma)^{\mathrm{op}} \to \mathbf{Set}$. This is to say that the projection π is a *representing object* for the functor M [7].

3 Categories with families

Now, we turn to the general idea of a model of type theory. There are many frameworks for defining such models, but we will opt for Dybjer's *categories with families* [2].

In fact, we've already seen our first example of a category with families—this was the content of §2. This model is known as the *syntactic model* or *term*

model. Keeping the syntactic model in mind, hopefully the following definition makes sense.

Definition 3.1. A *category with families* (CwF) consists of the following data:

- A *category of contexts* **Ctx** with a terminal object ().
- A *term functor* $\mathrm{Tm}\colon \mathbf{Ctx}^{\mathrm{op}} \to \mathbf{Fam}$. We will use $\mathrm{Ty}(\Gamma)$ to refer to the index set of $\mathrm{Tm}(\Gamma)$. For a substitution σ, we write $A\sigma$ instead of $\mathrm{Tm}(\sigma)(A)$, and $t\sigma$ instead of $\mathrm{Tm}(\sigma)(t)$.
- For each context $\Gamma \in \mathbf{Ctx}$ and type $A \in \mathrm{Ty}(\Gamma)$:
  - An *extended context* $(\Gamma,\, x : A)$;
  - A substitution π from $(\Gamma,\, x : A)$ to Γ, called the *projection*;
  - A term $q \in \mathrm{Tm}\big((\Gamma,\, x : A), A\pi\big)$;

  such that for any context Δ, substitution σ from Δ to Γ, and term $t \in \mathrm{Tm}\big(\Delta, A\sigma\big)$, there is a unique *extended substitution* $\langle \sigma, t\rangle$ from Δ to $(\Gamma,\, x : A)$ satisfying

$$\pi\langle \sigma, t\rangle = \sigma \qquad\qquad q\langle \sigma, t\rangle = t$$

To actually turn a CwF into a fully fledged model of a given type theory, we have to explain how to interpret all of the theory's type- and term-formers (e.g. $\to$, Π, Σ, $=$, λ, inductive types, etc). We will do this in more detail for our next example.

4 Set model

Let's see our second example of a category with families—the *set-theoretic model* or *set model* [1]. This could be considered the "classical" model of type theory. The definition of the set model will be informal, and we'll leave it to the reader to formalise it in the style of Definition 3.1.

For every piece of syntax S, we give it an interpretation $[\![S]\!]$ as follows. Given a context

$$\begin{aligned}\Gamma = \big(\; & x_1 : A_1,\\ & x_2 : A_2(x_1),\\ & x_3 : A_3(x_1, x_2),\\ & \quad\vdots\\ & x_n : A_n(x_1, \ldots, x_{n-1})\;\big)\end{aligned}$$

the type $A_i(x_1, \ldots, x_{i-1})$ is interpreted as a family of sets *indexed by the previous types*. You could also think of this as a *dependent function* of the previous types. So, we have:

- A set $[\![A_1]\!]$;
- For each $a_1 \in [\![A_1]\!]$, a set $[\![A_2]\!](a_1)$;
- For each $a_1 \in [\![A_1]\!]$ and $a_2 \in [\![A_2]\!](a_1)$, a set $[\![A_3]\!](a_1, a_2)$;

and so on.

Then, Γ is interpreted as the set $[\![\Gamma]\!]$ of *possible values* for its variables, i.e.

$$[\![\Gamma]\!] \coloneqq \left\{ (a_1, \ldots, a_n) \;\middle|\; a_i \in [\![A_i]\!](a_1, \ldots, a_{i-1}) \right\}$$

The empty context is interpreted by a singleton set $[\![()]\!] \coloneqq \{*\}$.

A substitution σ from Δ to Γ is interpreted as a set-theoretic function $[\![\sigma]\!] \colon [\![\Delta]\!] \to [\![\Gamma]\!]$, and composition is the usual function composition.

For every context Γ, a *type A over Γ* is a *family of sets* $[\![A]\!]$ indexed by $[\![\Gamma]\!]$. That is, for every possible assignment of variables $(a_1, \ldots, a_n) \in [\![\Gamma]\!]$, we have a set $[\![A]\!](a_1, \ldots, a_n)$. So the collection of types $\mathrm{Ty}(\Gamma)$ over Γ is interpreted as

$$[\![\mathrm{Ty}(\Gamma)]\!] \coloneqq \left\{ [\![A]\!] \colon [\![\Gamma]\!] \to \mathbf{Set} \right\}$$

Given a type $A \in \mathrm{Ty}(\Gamma)$, a term $t : A$ is interpreted as a *dependent function* $[\![t]\!]$ sending each $(a_1, \ldots, a_n) \in [\![\Gamma]\!]$ to an element $[\![t]\!](a_1, \ldots, a_n) \in [\![A]\!](a_1, \ldots, a_n)$. The collection of terms $\mathrm{Tm}(\Gamma, A)$ is interpreted as the set $\big[\!\big[\mathrm{Tm}(\Gamma, A)\big]\!\big]$ of all such dependent functions.

Now, given a substitution $\sigma \colon \Gamma \to \Delta$, the induced action on types $B \in \mathrm{Ty}(\Delta)$ is

$$[\![B\sigma]\!] \colon\ (a_1, \ldots, a_n) \ \mapsto\ [\![B]\!]\big(\sigma(a_1, \ldots, a_n)\big)$$

and on terms $t \in \mathrm{Tm}(\Delta, B)$:

$$[\![t\sigma]\!] \ :\ (a_1, \ldots, a_n) \ \mapsto\ [\![t]\!]\big(\sigma(a_1, \ldots, a_n)\big)$$

Given a context Γ and a type $A \in \mathrm{Ty}(\Gamma)$, the extended context $(\Gamma,\, x : A)$ is interpreted as the set

$$[\![(\Gamma,\, x : A)]\!] \coloneqq \left\{ (a_1, \ldots, a_n, a) \;\middle|\; (a_1, \ldots, a_n) \in [\![\Gamma]\!],\ a \in [\![A]\!](a_1, \ldots, a_n) \right\}$$

The projection π from $(\Gamma,\, x : A)$ to Γ is interpreted simply as the set-theoretic projection

$$\begin{array}{rrcl} [\![\pi]\!] \colon & [\![(\Gamma,\, x : A)]\!] & \to & [\![\Gamma]\!] \\ [\![\pi]\!] \colon & (a_1, \ldots, a_n, a) & \mapsto & (a_1, \ldots, a_n) \end{array}$$

while the second projection q is interpreted as the term $[\![q]\!] \in \big[\!\big[\mathrm{Tm}\big((\Gamma,\, x : A), A\pi\big)\big]\!\big]$ sending $(a_1, \ldots, a_n, a)$ to a.

Given a substitution $\sigma\colon \Delta \to \Gamma$ and term $t \in \mathrm{Tm}\big(\Delta, A\sigma\big)$, we interpret the extended substitution $\langle \sigma, t\rangle$ as

$$\begin{aligned}
[\![\langle \sigma, t\rangle]\!]\colon \quad & [\![\Delta]\!] && \to && [\![(\Gamma,\, x : A)]\!] \\
[\![\langle \sigma, t\rangle]\!]\colon \quad & (b_1, \ldots, b_m) && \mapsto && \Big([\![\sigma]\!](b_1, \ldots, b_m),\, [\![t]\!](b_1, \ldots, b_m)\Big)
\end{aligned}$$

We leave it to the reader to verify the universal property of $[\![\langle \sigma, t\rangle]\!]$.

At this point, we have fully defined a category with families. For the interested reader, we show in Appendix A how to interpret Π-types, Σ-types and equality in the set model.

5 Groupoid model

With the set model under our belt, let's move on to another important model—the *groupoid model*. Historically, this model is important because it was the first model of Martin-Löf type theory in which *uniqueness of identity proofs* (UIP) failed—that is, we can find a type A and elements $a, b : A$ such that $(a = b)$ contains more than one distinct element.

The idea is fairly simple, actually. It has long been observed that the equalities on a type A form a *groupoid* structure on A. Hofmann and Streicher's idea was to *interpret* types as groupoids. Then, we can show UIP fails by exhibiting a groupoid with two distinct parallel paths (of which there are many). It is a bit more complex to turn this idea into a full model of MLTT, but we will show how in this section.

The groupoid model is essentially built by "adding extra structure" to the set model, so we recommend fully understanding §4 before reading this one. For completeness, we'll start by defining groupoids.

Definition 5.1. A groupoid is a category where every morphism has an inverse. More concretely, a groupoid $\mathcal{G}$ consists of:

- A set $\mathrm{ob}\,\mathcal{G}$ of *objects of* $\mathcal{G}$;
- For every $a, b \in \mathrm{ob}\,\mathcal{G}$, a set $\hom(a, b)$ of *morphisms from* a *to* b;
- For every $a \in \mathrm{ob}\,\mathcal{G}$, an *identity morphism* $\mathrm{id}_a \in \hom(a, a)$;
- For every $a, b \in \mathrm{ob}\,\mathcal{G}$, an *inverse function* $(\cdot)^{-1}\colon \hom(a, b) \to \hom(b, a)$;
- For every $a, b, c \in \mathrm{ob}\,\mathcal{G}$, a *composition operation* $\circ\colon \hom(b, c) \times \hom(a, b) \to \hom(a, c)$;

such that:

- For every $f \in \hom(a, b)$, $f \circ \mathrm{id}_a = f$ and $\mathrm{id}_b \circ f = f$;
- For every $f \in \hom(a, b)$, $f^{-1} \circ f = \mathrm{id}_a$ and $f \circ f^{-1} = \mathrm{id}_b$;

- For every $f \in \hom(a, b)$, $g \in \hom(b, c)$ and $c \in \hom(c, d)$, $h \circ (g \circ f) = (h \circ g) \circ f$.

In the context of the groupoid model, a groupoid $\mathcal{G}$ is best understood as a set ($\operatorname{ob}\mathcal{G}$) with extra "equalities" (morphisms) imposed on top, where two objects a, b are "equal" if there is a morphism in $\hom(a, b)$.

A *groupoid homomorphism* from $\mathcal{G}$ to $\mathcal{H}$ is just a functor from $\mathcal{G}$ to $\mathcal{H}$, when viewed as categories. Groupoids and groupoid homomorphisms therefore form a full subcategory **Gpd** of **Cat**.

To model *dependent* types using groupoids, we need a notion of a family of groupoids dependent on/indexed by another groupoid. Henceforth, we will abuse notation slightly, and identify a groupoid with its underlying set of objects. Thus, we will use $a \in \mathcal{G}$ to mean $a \in \operatorname{ob}\mathcal{G}$.

Definition 5.2. Given a groupoid $\mathcal{G}$, a *family of groupoids indexed over* $\mathcal{G}$ is a functor $\mathcal{G} \to \mathbf{Gpd}$. More concretely, it consists of:

- For every $a \in G$, a groupoid $\mathcal{H}_a$;
- For every morphism $p \in \hom(a, b)$, a groupoid homomorphism $F_p \colon \mathcal{H}_a \to \mathcal{H}_b$;

so that F_{id} is the identity homomorphism, and $F_p \circ F_q = F_{(p \circ q)}$.

The groupoid axioms ensure that each F_p is an isomorphism of groupoids. The intended meaning is that if $a = b$ in $\mathcal{G}$, then the groupoids $\mathcal{H}_a$ and $\mathcal{H}_b$ are isomorphic.

Now, we're ready to define the groupoid model. For every piece of syntax S, we give it an interpretation $[\![S]\!]$ as follows. Given a context

$$\begin{aligned}\Gamma = (\ & x_1 : A_1, \\ & x_2 : A_2(x_1), \\ & x_3 : A_3(x_1, x_2), \\ & \quad \vdots \\ & x_n : A_n(x_1, \ldots, x_{n-1})\ )\end{aligned}$$

the type $A_i(x_1, \ldots, x_{i-1})$ is interpreted as a family of *groupoids* indexed by the previous types. As in the set model, we have:

- A groupoid $[\![A_1]\!]$;
- For each $a_1 \in [\![A_1]\!]$, a groupoid $[\![A_2]\!](a_1)$;
- For each $a_1 \in [\![A_1]\!]$ and $a_2 \in [\![A_2]\!](a_1)$, a groupoid $[\![A_3]\!](a_1, a_2)$;

and so on. But, additionally, the equalities in earlier types induce *coherences* in the later types:

- For each $[\![A_1]\!]$-morphism $p_1 \in \hom(a_1, b_1)$,
  we get a groupoid isomorphism $\overline{p_1} \colon [\![A_2]\!](a_1) \to [\![A_2]\!](b_1)$;
- For each $[\![A_1]\!]$-morphism $p_1 \in \hom(a_1, b_1)$
  and $[\![A_2]\!](b_1)$-morphism $p_2 \in \hom(\overline{p_1}a_2, b_2)$,
  we get a groupoid isomorphism $\overline{p_1 p_2} \colon [\![A_3]\!](a_1, a_2) \to [\![A_3]\!](b_1, b_2)$;

and so on.

As in the set model, Γ is interpreted as the set $[\![\Gamma]\!]$ of *possible values* for its variables:

$$[\![\Gamma]\!] \coloneqq \Big\{ (a_1, \ldots, a_n) \;\Big|\; a_i \in [\![A_i]\!](a_1, \ldots, a_{i-1}) \Big\}$$

The morphisms of $[\![\Gamma]\!]$ are those induced by the groupoid structure on each of the $[\![A_i]\!]$:

$$\begin{aligned} &\hom_{[\![\Gamma]\!]}\big((a_1, \ldots, a_n), (b_1, \ldots, b_n)\big) \\ &\qquad \coloneqq \big\{ (p_1, \ldots, p_n) \;\big|\; p_i \in \hom(\overline{p_1 \cdots p_{i-1}}a_i, b_i) \big\} \end{aligned}$$

The empty context is interpreted by the trivial groupoid with one object and one identity morphism, as shown on the right.

A substitution σ from Δ to Γ is interpreted as a groupoid homomorphism $[\![\sigma]\!] \colon [\![\Delta]\!] \to [\![\Gamma]\!]$.

Henceforth, we will abbreviate $(a_1, \ldots, a_n) \in [\![\Gamma]\!]$ as $\overline{a}$. For a context Γ, a *type* A *over* Γ is a family of *groupoids* $[\![A]\!]$ indexed by $[\![\Gamma]\!]$. That is, for every $\overline{a} \in [\![\Gamma]\!]$, we have a groupoid $[\![A]\!](\overline{a})$, and for every $[\![\Gamma]\!]$-morphism $p \in \hom(\overline{a}, \overline{b})$, a groupoid isomorphism $[\![A]\!](p) \colon [\![A]\!](\overline{a}) \to [\![A]\!](\overline{b})$.

For a type A over Γ, a term $s \in \mathrm{Tm}(\Gamma, A)$ is interpreted as a "dependent functor" $[\![\Gamma]\!] \to [\![A]\!]$:

- For each $\overline{a} \in [\![\Gamma]\!]$, we have an object $[\![s]\!](\overline{a}) \in [\![A]\!](\overline{a})$;
- For each $[\![\Gamma]\!]$-morphism $p \in \hom(\overline{a}, \overline{b})$, an $[\![A]\!](\overline{b})$-morphism

$$[\![s]\!](p) \;\in\; \hom\Big([\![A]\!](p)\big([\![s]\!](\overline{a})\big),\, [\![s]\!](\overline{b})\Big)$$

Given a substitution $\sigma \colon \Gamma \to \Delta$ and a type $B \in \mathrm{Ty}(\Delta)$, we interpret $B\sigma \in \mathrm{Ty}(\Gamma)$ as follows. $[\![B\sigma]\!]$ sends:

- $\overline{a} \in [\![\Gamma]\!]$ to the groupoid $[\![B]\!](\sigma(\overline{a}))$;
- $p \in \hom_{[\![\Gamma]\!]}(\overline{a}, \overline{b})$ to the homomorphism $[\![B]\!](\sigma(p)) \colon [\![B]\!](\sigma(\overline{a})) \to [\![B]\!](\sigma(\overline{b}))$.

Similarly, for $t \in \mathrm{Tm}(\Delta, B)$, we interpret $t\sigma \in \mathrm{Tm}(\Gamma, B\sigma)$ as the map $[\![t\sigma]\!]$ sending:

- $\overline{a} \in [\![\Gamma]\!]$ to the object $[\![t]\!](\sigma(\overline{a})) \in [\![B]\!](\sigma(\overline{a}))$;

- $p \in \hom_{[\![\Gamma]\!]}(\overline{a}, \overline{b})$ to the $[\![B]\!](\sigma(\overline{b}))$-morphism

$$[\![t]\!](\sigma(p)) \;\in\; \hom\Big([\![B]\!](\sigma(p))\big([\![t]\!](\sigma(\overline{a}))\big),\, [\![t]\!](\sigma(\overline{b}))\Big)$$

Given a context Γ and a type $A \in \mathrm{Ty}(\Gamma)$, the extended context $(\Gamma,\, x : A)$ is interpreted as the groupoid with objects

$$[\![\Gamma,\, x : A]\!] \coloneqq \big\{(\overline{a}, a) \;\big|\; \overline{a} \in [\![\Gamma]\!],\; a \in [\![A]\!](\overline{a})\big\}$$

and morphisms

$$\hom_{[\![\Gamma,\, x:A]\!]}\big((\overline{a}, a), (\overline{b}, b)\big) \;\coloneqq\; \Big\{(p, p') \;\Big|\; \begin{array}{l} p \in \hom_{[\![\Gamma]\!]}(\overline{a}, \overline{b}), \\ \qquad p' \in \hom_{[\![A]\!](\overline{b})}\big([\![A]\!](p)(a),\, b\big) \end{array}\Big\}$$

The projection π from $(\Gamma,\, x : A)$ to Γ is interpreted as the groupoid homomorphism sending $(\overline{a}, a)$ to $\overline{a}$, and (p, p') to p. Similarly, the second projection q is interpreted as the term $[\![q]\!] \in [\![\mathrm{Tm}\big((\Gamma,\, x : A), A\pi\big)]\!]$ sending $(\overline{a}, a)$ to a, and (p, p') to p'.

Finally, given a substitution $\sigma \colon \Delta \to \Gamma$ and term $t \in \mathrm{Tm}\big(\Delta, A\sigma\big)$, we interpret the extended substitution $\langle \sigma, t \rangle : \Delta \to (\Gamma,\, x : A)$ as follows:

- $[\![\langle \sigma, t \rangle]\!]$ maps $\overline{c} \in [\![\Delta]\!]$ to $\big([\![\sigma]\!](\overline{c}), [\![t]\!](\overline{c})\big) \in [\![\Gamma,\, x : A]\!]$.
- $[\![\langle \sigma, t \rangle]\!]$ maps $p \in \hom_{[\![\Delta]\!]}(\overline{c}, \overline{d})$ to $\big([\![\sigma]\!](p), [\![t]\!](p)\big)$.

5.1 Equality

We have now fully defined a category with families. We will leave the interpretation of Π-types and Σ-types to Appendix B. However, we will talk here about *equality types*, since they are the original motivation (and arguably, the defining feature) of the groupoid model.

finish this

6 Conclusion

other models such as setoid model higher groupoid model presheaf model sheaf model

References

[1] Thierry Coquand. *What is a model of type theory?* Notes from the Special Year on Univalent Foundations of Mathematics, Institute for Advanced Study, Princeton, 2012. `https://ncatlab.org/ufias2012/files/cwf1.pdf`

[2] Peter Dybjer. *Internal type theory.* pp120–134 in: Stefano Berardi, Mario Coppo (eds.), *Types for Proofs and Programs* (proceedings from TYPES '95, Torino, Italy, June 1995). Lecture Notes in Computer Science **1158**, Springer, 1996. Available online at `https://www.cse.chalmers.se/~peterd/papers/InternalTT.pdf`

[3] Martin Hofmann, Thomas Streicher. *The groupoid interpretation of type theory.* pp83–112 in: Giovanni Sambin, Jan M. Smith (eds.), *Twenty Five Years of Constructive Type Theory* (conference proceedings from Venice, October 1995). Oxford Logic Guides **36**, Oxford University Press, 1998.

[4] Tom Leinster. *Basic Category Theory.* Cambridge Studies in Advanced Mathematics **143**, Cambridge University Press, 2014. `arXiv:1612.09375`

[5] Saunders Mac Lane. *Categories for the Working Mathematician.* 2nd ed. Graduate Texts in Mathematics **5**, Springer, 1998.

[6] *Homotopy Type Theory: Univalent Foundations of Mathematics* (a.k.a. "the HoTT book"). 2013. Available at `https://homotopytypetheory.org/book/`

[7] The nLab. *Categorical model of dependent types > Categories with families.* `https://ncatlab.org/nlab/show/categorical+model+of+dependent+types#categories_with_families`

[8] The nLab. *Syntactic category.* `https://ncatlab.org/nlab/show/syntactic+category`

A More on the set model

In this section, we show how to interpret some of the common type- and term-formers in the set model defined in §4. Hence, this should be read directly after that section. Henceforth, we will abbreviate $(a_1, \dots, a_n) \in [\![\Gamma]\!]$ as $\overline{a}$.

A.1 Π-types

Given a context Γ and types $A, B \in \mathrm{Ty}(\Gamma)$, the arrow type $(A \to B) \in \mathrm{Ty}(\Gamma)$ is interpreted as the family

$$[\![A \to B]\!](\overline{a}) \;\coloneqq\; \big\{\text{functions from } [\![A]\!](\overline{a}) \text{ to } [\![B]\!](\overline{a})\big\}$$

More generally, if $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}\big((\Gamma,\, x : A)\big)$, then $\prod_{(x:A)} B(x) \in \mathrm{Ty}(\Gamma)$ is interpreted as

$$\left[\!\!\left[\prod_{(x:A)} B(x)\right]\!\!\right](\overline{a}) \;\coloneqq\; \Big\{\text{functions sending each } a \in [\![A]\!](\overline{a}) \text{ to an element of } [\![B]\!](\overline{a}, a)\Big\}$$

Given a term $t \in \mathrm{Tm}\big((\Gamma,\, x : A), B\big)$, the lambda abstraction $(\lambda(x : A) \mapsto t) \in \mathrm{Tm}\big(\Gamma, \prod_{(x:A)} B(x)\big)$ is interpreted as

$$\big[\!\big[\lambda(x : A) \mapsto t\big]\!\big](\overline{a}) \;\coloneqq\; a \mapsto [\![t]\!](\overline{a}, a)$$

Now, given a term $s \in \mathrm{Tm}(\Gamma, A)$, we write $B(s) \coloneqq B(\mathrm{id}, s) \in \mathrm{Ty}(\Gamma)$, where $\langle \mathrm{id}, s\rangle$ is the extended substitution $\Gamma \to (\Gamma,\, x : A)$. Given another term $f \in \mathrm{Tm}\big(\Gamma, \prod_{(x:A)} B(x)\big)$, the application $f(s) \in \mathrm{Tm}\big(\Gamma, B(s)\big)$ is interpreted as

$$\big[\!\big[f(s)\big]\!\big](\overline{a}) \;\coloneqq\; [\![f]\!](\overline{a})\big([\![s]\!](\overline{a})\big)$$

Letting $t(s) \coloneqq t\langle \mathrm{id}, s\rangle \in \mathrm{Ty}\big(\Gamma, B(s)\big)$, the computation rule

$$(\lambda(x : A) \mapsto t)(s) \;\equiv\; t(s)$$

holds since both sides are interpreted as $\overline{a} \mapsto [\![t]\!]\big(\overline{a}, [\![s]\!](\overline{a})\big)$.

Given $f \in \mathrm{Tm}\big(\Gamma, \prod_{(x:A)} B(x)\big)$, write $f(x) \in \mathrm{Tm}\big((\Gamma,\, x : A), B\big)$ for the term interpreted

$$\big[\!\big[f(x)\big]\!\big](\overline{a}, a) \;\coloneqq\; [\![f]\!](\overline{a})(a)$$

Then $\big[\!\big[\lambda(x : A) \mapsto f(x)\big]\!\big] = [\![f]\!]$, validating the uniqueness rule

$$\big(\lambda(x : A) \mapsto f(x)\big) \equiv f$$

A.2 Σ-types

Given a context Γ and types $A, B \in \mathrm{Ty}(\Gamma)$, the product type $(A \times B) \in \mathrm{Ty}(\Gamma)$ is interpreted as the family

$$\big[\!\big[A \times B\big]\!\big](\overline{a}) \;\coloneqq\; [\![A]\!](\overline{a}) \times [\![B]\!](\overline{a})$$

More generally, if $A \in \mathrm{Ty}(\Gamma)$ and $B \in \mathrm{Ty}\big((\Gamma,\, x : A)\big)$, then $\sum_{(x:A)} B(x) \in \mathrm{Ty}(\Gamma)$ is interpreted as

$$\left[\!\!\left[\sum_{(x:A)} B(x)\right]\!\!\right](\overline{a}) \;\coloneqq\; \big\{(a, b) \;\big|\; a \in [\![A]\!](\overline{a}),\; b \in [\![B]\!](\overline{a}, a)\big\}$$

Given terms $s \in \mathrm{Tm}(\Gamma, A)$ and $t \in \mathrm{Tm}\big((\Gamma,\, x : A), B\big)$, the pair $(s, t) \in \mathrm{Tm}\Big(\Gamma, \sum_{(x:A)} B(x)\Big)$ is interpreted as

$$\big[\!\big[(s, t)\big]\!\big](\overline{a}) \;\coloneqq\; \Big([\![s]\!](\overline{a}),\; [\![t]\!]\big(\,\overline{a},\, [\![s]\!](\overline{a})\,\big)\Big)$$

Interpret projections, validate computation rule

A.3 Equality

Perhaps the distinguishing feature of the set model is its treatment of equality types. Unlike in other models, equality here behaves exactly like a classical mathematician would expect.

Given a type $A \in \mathrm{Ty}(\Gamma)$, the equality type $(x =_A y) \in \mathrm{Ty}\big((\Gamma,\, x : A,\, y : A\pi)\big)$ is interpreted as

$$[\![x =_A y]\!](\overline{a}, b, c) \coloneqq \begin{cases} \{*\} & b = c \\ \varnothing & b \neq c \end{cases}$$

Given $s \in \mathrm{Tm}(\Gamma, A)$, we interpret $\mathsf{refl}_s \in \mathrm{Ty}\big((\Gamma,\, x : A,\, y : A\pi), (s =_A s)\big)$?? as $[\![\mathsf{refl}_s]\!](\overline{a}) = *$.

Check rules

B More on the groupoid model

In this section, we show how to interpret Π-types and Σ-types in the groupoid model, as defined in §5. This section should be read after §4, §5 and Appendix A. As in the previous appendix, we will abbreviate $(a_1, \ldots, a_n) \in [\![\Gamma]\!]$ as $\overline{a}$.

B.1 Π-types

finish this

B.2 Σ-types

finish this