Lecture 1: Introduction and non-homotopical semantics

Homotopical semantics of type theory

Evan Cavallo

26 January 2026

This course is meant to be an overview, from one particular angle, of the *homotopical semantics* of *type theory*, and in particular the relationship between models of certain type theories and $(\infty, 1)$-*categories*, which are objects one works with in homotopy theory. This relationship is one of the key motivations for homotopy type theory: it means that we can use type theory to "work inside of" and prove things about individual $(\infty, 1)$-categories (such as the $(\infty, 1)$-category of topological spaces), and it means we can learn about what is provable in type theory by studying these models.

We are going to talk about multiple different type theories, all of which will be fragments or extensions of Martin-Löf's intensional type theory [Mar75]. My hope with this course is to say something precise about how models of these theories correspond to different classes of $(\infty, 1)$-categories. Even getting to precise theorem statements takes time, and proving the theorems is of course even harder! Indeed, this is still an area of active research, and we shall see that for fully-featured type theories, only partial results are known.

It is not too hard to get an intuitive feeling that there is a connection between type theory and homotopy theory. The basis for the connection is Martin-Löf's intensional identity type with the J eliminator, specified by the following rules:

$$\textsc{Formation}\quad \frac{M_0, M_1 : A}{\mathsf{Id}_A(M_0, M_1)\ \mathsf{type}} \qquad \textsc{Introduction}\quad \frac{M : A}{\mathsf{refl}_M : \mathsf{Id}_A(M, M)}$$

$$\textsc{Elimination (J)}\quad \frac{\begin{array}{c} a_0, a_1 : A, p : \mathsf{Id}_A(a_0, a_1) \vdash C(a_0, a_1, p)\ \mathsf{type} \\ a : A \vdash R(a) : C(a, a, \mathsf{refl}_a) \qquad M_0, M_1 : A \qquad P : \mathsf{Id}_A(M_0, M_1) \end{array}}{\mathsf{J}(C, R, M_0, M_1, P) : C(M_0, M_1, P)}$$

$$\textsc{Computation}\quad \frac{a_0, a_1 : A, p : \mathsf{Id}_A(a_0, a_1) \vdash C(a_0, a_1, p)\ \mathsf{type} \qquad a : A \vdash R(a) : C(a, a, \mathsf{refl}_a) \qquad M : A}{\mathsf{J}(C, R, M, M, \mathsf{refl}_M) = R(M) : C(M, M, \mathsf{refl}_M)}$$

The propositions-as-types philosophy puts identity types on the same level as "data" types like the natural numbers, and the elimination rule does not preclude a *proof-relevant* interpretation of identity types, where identity might be a structure on two elements rather than a property. Homotopy theory deals exactly with situations where "equality" is treated as structure, so we should expect to be able to analyze models of these rules using homotopy-theoretic tools.

However, coming up with the right statement to formalize this connection is not so easy. In this first lecture, we will not talk about homotopy theory at all, but rather review the relationship between models of *extensional type theory* and *1-categories*. This will help us to understand what we might want to prove in the intensional case.

1 Type theory

Let us recall the basic ingredients of type theory in a bit more detail, so that we can decide what we mean by a *model of type theory*. The statements we make in type theory are called *judgments*, and more specifically we make *type* and *term* judgments: we can state the existence of a type or the existence of a term of a given type.

$$A\ \mathsf{type} \qquad\qquad M : A$$

We also speak of equalities between the subjects of these judgments:

$$A_0 = A_1\ \mathsf{type} \qquad\qquad M_0 = M_1 : A$$

We moreover make *hypothetical judgments*, relative to a *context* Γ of assumptions, which we write like so:

$$\Gamma \vdash A\ \mathsf{type} \qquad\qquad \Gamma \vdash M : A$$

$$\Gamma \vdash A_0 = A_1\ \mathsf{type} \qquad\qquad \Gamma \vdash M_0 = M_1 : A$$

A type theory is then made up of rules that we apply to derive individual judgments.

In Martin-Löf's original presentations of type theory, a context is explicitly a list of hypotheses, each of which introduces a term variable of some type:

$$a_0 : A_0, a_1 : A_1(a_0), a_2 : A_2(a_0, a_1), \ldots, a_n : A_n(a_0, \ldots, a_{n-1})$$

However, it is often more convenient to include a separate judgment form for contexts. In this way of doing things, we can still build up contexts by adding term hypotheses:

$$\frac{}{\cdot\ \mathsf{ctx}} \qquad\qquad \frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash A\ \mathsf{type}}{\Gamma.A\ \mathsf{ctx}}$$

but we write the rules of our theory without assuming that every context has been built up using these rules. We can avoid giving our hypotheses names, as in the notation "$\Gamma.A$" I used above, by introducing a judgment for *explicit substitutions* $\Gamma' \vdash \gamma : \Gamma$ [ACCL91; Mar92] that act on types and terms:

$$\frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma \vdash A\ \mathsf{type}}{\Gamma' \vdash A[\gamma]\ \mathsf{type}} \qquad\qquad \frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma \vdash M : A}{\Gamma' \vdash M[\gamma] : A[\gamma]}$$

Access to variables from the context is provided by a variable term q_A, whose typing rule makes use of the projection or weakening substitution p_A:

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A \vdash \mathsf{p}_A : \Gamma} \qquad\qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma.A \vdash \mathsf{q}_A : A[\mathsf{p}_A]}$$

The term q_A is the variable corresponding to the rightmost hypothesis in the context. To access other variables, we use p and q together, for example $\Gamma.A.B \vdash \mathsf{q}_A[\mathsf{p}_B] : A[\mathsf{p}_A][\mathsf{p}_B]$. We build up substitutions in the same way that we build up contexts, by iterated extension:

$$\frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma' \vdash M : A[\gamma]}{\Gamma' \vdash \gamma.M : \Gamma.A}$$

Although we would be able to get type theory off the ground without doing so, we usually include *identity* and *composition* substitutions, which make the contexts and substitutions between

them into a category(!):

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathsf{id}_\Gamma : \Gamma} \qquad \frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma'' \vdash \gamma' : \Gamma'}{\Gamma'' \vdash \gamma \circ \gamma' : \Gamma} \qquad \frac{\Gamma' \vdash \gamma : \Gamma}{\Gamma' \vdash \mathsf{id}_\Gamma \circ \gamma = \gamma : \Gamma} \qquad \frac{\Gamma' \vdash \gamma : \Gamma}{\Gamma' \vdash \gamma \circ \mathsf{id}_{\Gamma'} = \gamma : \Gamma}$$

$$\frac{\dots}{\Gamma''' \vdash (\gamma \circ \gamma') \circ \gamma'' = \gamma \circ (\gamma' \circ \gamma'') : \Gamma}$$

We also have equations for the interaction of substitution extension with projection and the variable:

$$\frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma' \vdash M : A[\gamma]}{\Gamma' \vdash \mathsf{p}_A \circ (\gamma.M) = \gamma : \Gamma} \qquad \frac{\Gamma' \vdash \gamma : \Gamma \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma' \vdash M : A[\gamma]}{\Gamma' \vdash \mathsf{q}_A[\gamma.M] = M : A[\gamma]}$$

Exercise 1.1. Pick your favorite type former (*e.g.*, Σ, Π, or the Id-types from above) and write its rules using explicit substitutions instead of named variables.

For a modern introduction to type theory in this style, see Angiuli and Gratzer's textbook-in-progress [AG25, §2.3].

Remark 1.2. In this course, we are not going to work *formally* with syntax in the traditional sense. We will define formally what a *model of type theory (with,* e.g.*,* Σ *and* Id *types)* is, and we think of the sequents and inference rules above as *informal* notation for objects we have and constructions we can perform in any model. Partly for this reason, I won't shy from sacrificing precision for readability when I write rules; if you want to be sure of what I really mean, look at the definition of model.

A definition of model and a formally defined syntax can be related by an *initiality theorem*, as for example in Streicher [Str91, Chapter 4], De Boer [Boe20], or Uemura [Uem21, Chapter 5], but we will stay away from these matters in this course.

2 Models of type theory

We will use Awodey's *natural model* [Awo16] definition of model of type theory. Natural models are interchangeable with Cartmell's *categories with attributes* [Car78, §3.2; Pit01, §6.3] and Dybjer's *categories with families* [Dyb96]; the natural model presentation has certain advantages for the categorical study of models. For a recent summary of the relationships between the many different definitions of "model of type theory" in the literature, see Ahrens, Lumsdaine, and North [ALN24].

Notation 2.1. I write $\mathrm{PSh}(\mathcal{C})$ for the category of presheaves on a category $\mathcal{C}$, and $\text{よ}\colon \mathcal{C} \to \mathrm{PSh}(\mathcal{C})$ for the Yoneda embedding, which sends an object $\Gamma \in \mathcal{C}$ to the *representable* presheaf $\text{よ}\Gamma := \mathcal{C}(-, \Gamma)$.

Remark 2.2. We will use presheaf categories and the Yoneda embedding in many contexts in this course. If you need to familiarize yourself, Awodey [Awo06, Chapter 8 and Example 9.22] and Mac Lane and Moerdijk [MM94, Chapter I] cover many basic properties.

Definition 2.3. In a presheaf category $\mathrm{PSh}(\mathcal{C})$, a morphism $p\colon K \to J$ is *(locally) representable* when for every $\Gamma \in \mathcal{C}$ and $A\colon \text{よ}\Gamma \to J$, there is an object $\Gamma.A \in \mathcal{C}$ and morphisms $p_A\colon \Gamma.A \to \Gamma$ and $q_A\colon \text{よ}(\Gamma.A) \to K$ fitting into a pullback square

$$\begin{array}{ccc} \text{よ}(\Gamma.A) & \xrightarrow{q_A} & K \\ {\scriptstyle p_A}\downarrow & & \downarrow{\scriptstyle p} \\ \text{よ}\Gamma & \xrightarrow[A]{} & J \end{array}$$

Definition 2.4. A *natural model of type theory* $(\mathcal{C}, \pi)$ is a category $\mathcal{C}$ with a terminal object $1 \in \mathcal{C}$, two presheaves $\mathrm{Ty}, \mathrm{Tm} \in \mathrm{PSh}(\mathcal{C})$, and a representable map $\pi\colon \mathrm{Tm} \to \mathrm{Ty}$ between them.

We understand $\mathcal{C}$ as the semantic category of contexts and substitutions. For every object $\Gamma \in \mathcal{C}$, we have a set Ty_Γ of semantic types in context Γ and a set Tm_Γ of semantic terms in context; the map $\pi_\Gamma \colon \mathrm{Ty}_\Gamma \to \mathrm{Tm}_\Gamma$ sends each term to its type. For each $\gamma \colon \Gamma' \to \Gamma$, we get a function $\mathrm{Ty}_\gamma \colon \mathrm{Ty}_\Gamma \to \mathrm{Ty}_{\Gamma'}$ which we think of as mapping $\Gamma \vdash A\ \mathsf{type}$ to $\Gamma' \vdash A[\gamma]\ \mathsf{type}$ and a function $\mathrm{Tm}_\gamma \colon \mathrm{Tm}_\Gamma \to \mathrm{Tm}_{\Gamma'}$ which we think of as mapping $\Gamma \vdash M : A$ to $\Gamma' \vdash M[\gamma] : A[\gamma]$.

The Yoneda lemma identifies elements $A \in \mathrm{Ty}_\Gamma$ with presheaf morphisms $A \colon \yo\Gamma \to \mathrm{Ty}$, which lets us conveniently think of a type in context Γ as a "function" from Γ into the collection Ty of types; the same goes for Tm_Γ and $\yo\Gamma \to \mathrm{Tm}$. The condition that π is representable formalizes context extension: for every context $\Gamma \in \mathcal{C}$ and Γ-indexed family of types $A \colon \Gamma \to \mathrm{Ty}$, we get a new context $\Gamma.A \in \mathcal{C}$:

$$\begin{array}{ccc} \yo(\Gamma.A) & \xrightarrow{\mathsf{q}_A} & \mathrm{Tm} \\ {\scriptstyle \mathsf{p}_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi} \\ \yo\Gamma & \xrightarrow[A]{} & \mathrm{Ty}. \end{array}$$

This context has a projection substitution down to Γ (p_A), and there is a function from $\Gamma.A$ to terms $\mathsf{q}_A \colon \yo(\Gamma.A) \to \mathrm{Tm}$ that projects the hypothesized element of A. The fact that this square is a pullback means that a substitution into $\Gamma.A$ is given by a pair of a substitution into Γ and a term of type A.

2.1 Type formers

A natural model is a model of type theory without any type formers. To describe models of type theory with type formers, we ask for additional structure on $\pi \colon \mathrm{Tm} \to \mathrm{Ty}$. Often there is a nice categorical description of this structure. An simple example is extensional identity types, *i.e.*, identity types with the reflection rule (see §3):

$$\textsc{Reflection}\quad \frac{P : \mathsf{Id}_A(M_0, M_1)}{M_0 = M_1 : A}$$

Definition 2.5. An *extensional identity type structure* on a natural model $(\mathcal{C}, \pi)$ consists of morphisms $\mathsf{Id} \colon \mathrm{Tm} \times_{\mathrm{Ty}} \mathrm{Tm} \to \mathrm{Ty}$ and $\mathsf{refl} \colon \mathrm{Tm} \to \mathrm{Tm}$ such that we have a pullback square

$$\begin{array}{ccc} \mathrm{Tm} & \xrightarrow{\mathsf{refl}} & \mathrm{Tm} \\ {\scriptstyle \langle \mathrm{id}_{\mathrm{Tm}}, \mathrm{id}_{\mathrm{Tm}} \rangle}\downarrow & \lrcorner & \downarrow{\scriptstyle \pi} \\ \mathrm{Tm} \times_{\mathrm{Ty}} \mathrm{Tm} & \xrightarrow[\mathsf{Id}]{} & \mathrm{Ty}. \end{array}$$

However, we can also translate sequents and inference rules into the language of presheaves in a completely mechanical way. Say, for example, that we want to define Σ type structure. We can represent the hypotheses of its formation rule by a presheaf $\mathrm{Fam} \in \mathrm{PSh}(\mathcal{C})$:

$$\mathrm{Fam}_\Gamma \coloneqq \{(A, B) \mid A \in \mathrm{Ty}_\Gamma, B \in \mathrm{Ty}_{\Gamma.A}\} \qquad \mathrm{Fam}_\gamma(A, B) \coloneqq (A\gamma, B\langle \gamma \mathsf{p}_A, \mathsf{q}_A \rangle).$$

Then we can ask our natural model to interpret the Σ formation rule by asking for a morphism $\Sigma \colon \mathrm{Fam} \to \mathrm{Ty}$. For the introduction rule, we can similarly define a presheaf $\mathrm{Pair} \in \mathrm{PSh}(\mathcal{C})$ with a projection $\pi^{\mathrm{Pair}} \colon \mathrm{Pair} \to \mathrm{Fam}$:

$$\mathrm{Pair}_\Gamma \coloneqq \{(A, B, M, N) \mid A \in \mathrm{Ty}_\Gamma, B \in \mathrm{Ty}_{\Gamma.A}, M \in \pi_\Gamma^{-1}(A), N \in \pi_\Gamma^{-1}(B\langle \mathrm{id}_\Gamma, M \rangle)\}$$
$$\pi^{\mathrm{Pair}}(A, B, M, N) \coloneqq (A, B)$$

and ask for our model to interpret the introduction rule by asking for a morphism

$$\begin{array}{ccc} \mathrm{Pair} & \dashrightarrow & \mathrm{Tm} \\ {\scriptstyle \pi^{\mathrm{Pair}}}\downarrow & & \downarrow{\scriptstyle \pi} \\ \mathrm{Fam} & \xrightarrow[\Sigma]{} & \mathrm{Ty} \end{array}$$

Exercise 2.6. Formulate the remaining rules for Σ types in the language of natural models. Do Σ types have a characterization analogous to Definition 2.5?

3 Extensional type theory and category theory

In Martin-Löf's extensional type theory (ETT) [Mar82], identity types are trivialized by adding the *reflection* rule, which says that any identified terms are judgmentally equal:

$$\textsc{Reflection}\quad \frac{P : \mathsf{Id}_A(M_0, M_1)}{M_0 = M_1 : A}$$

Although this is not immediately obvious, the reflection rule forces equality to be proof-irrelevant:

Exercise 3.1. Given the formation (Id) and introduction (refl) rules for identity types together with the reflection rule, show that the J rule is interderivable with the rule

$$\textsc{Uniqueness}\quad \frac{M_0, M_1 : A}{P = \mathsf{refl}_{M_0} : \mathsf{Id}_A(M_0, M_1)}$$

There is a correspondence between "democratic" (see below) models of ETT with Σ, Π, and Id and *locally cartesian closed categories* (LCCC's). Seely [See84] made the first attempt at such a result, but there is a problem with his construction of a model of type theory from an LCCC. Curien [Cur93] and Hofmann [Hof95] gave two different corrected versions of that construction (see also [CGH14]). Using Hofmann's construction, Clairambault and Dybjer [CD14] proved a correct version of the correspondence: a biequivalence between a 2-category of democratic models of ETT and the established 2-category of LCCC's. Clairambault and Dybjer also observe that if we throw away Π types, we get a correspondence between models of ETT with Σ and Id and categories with finite limits.

3.1 Democratic models

From our definition of models in §2, we see right away that every model of type theory $(\mathcal{C}, \pi)$ contains a category: the category of contexts $\mathcal{C}$. At the same time, it seems a priori that $(\mathcal{C}, \pi)$ contains more data, including multiple other categories. For every context $\Gamma \in \mathcal{C}$, there is a category $\mathbf{Ty}_\Gamma$ whose objects are types $A : \mathfrak{F}\Gamma \to \mathrm{Ty}$ and whose morphisms $A \to B$ are substitutions $\Gamma.A \to \Gamma.B$ over Γ (or equivalently terms of type $B\mathsf{p}_A$ in context $\Gamma.A$). In particular, there is a category $\mathbf{Ty}_1$ of *closed types*. There is a fully faithful functor $1.(-) : \mathbf{Ty}_1 \to \mathcal{C}$ sending each closed type A to the context $1.A$, but this functor may not have an inverse.

When we relate models to categories, therefore, we restrict our attention to models where every context *does* arise from iterated context extension.

Definition 3.2. For a natural model $(\mathcal{C}, \pi : \mathrm{Tm} \to \mathrm{Ty})$, the class of *contextual objects* in $\mathcal{C}$ is the least replete class containing

(a) the terminal object $1 \in \mathcal{C}$,

(b) for every contextual object $\Gamma \in \mathcal{C}$ and $A\colon \text{ѫ}\Gamma \to \mathrm{Ty}$, the object $\Gamma.A \in \mathcal{C}$.

A natural model is *democratic* when every object of $\mathcal{C}$ is contextual.

For a natural model with Σ types, every contextual object is isomorphic to a context consisting of a single closed type, since $1.A.B \cong 1.\Sigma(A,B)$. This makes the functor $1.(-)\colon \mathbf{Ty}_1 \to \mathcal{C}$ into an equivalence.

As for the categories $\mathbf{Ty}_\Gamma$ for other Γ, we will see that in democratic models they are essentially determined by $\mathcal{C}$.

3.2 From models to finite limit categories

We now show that for a democratic model $(\mathcal{C}, \pi)$ with Σ and extensional Id types, the category of contexts $\mathcal{C}$ has finite limits. Following Clairambault and Dybjer, we prove this with the help of the following lemma:

Lemma 3.3. For every $\gamma\colon \Gamma' \to \Gamma$, there is a type $A\colon \text{ѫ}\Gamma \to \mathrm{Ty}$ with an isomorphism

$$\begin{array}{ccc} \Gamma' & \xrightarrow[\cong]{\theta} & \Gamma.A \\ & {\scriptstyle\gamma}\searrow \quad \swarrow{\scriptstyle \mathsf{p}_A} & \\ & \Gamma & \end{array}$$

in $\mathcal{C}$.

Proof. Because $(\mathcal{C}, \pi)$ is democratic, we have closed types $B, B' \in \mathbf{Ty}_1$ such that $\Gamma \cong 1.B$ and $\Gamma' \cong 1.B'$. Since a substitution between closed types consists of a term, it will suffice to show the statement for the special case where $\gamma = \langle \mathsf{p}_{B'}, t\rangle\colon 1.B' \to 1.B$ for some term $1.B' \vdash t : B$. Let us work informally using named variables, so $b' : B' \vdash t(b') : B$. We define a type $b : B \vdash A(b)\ \mathsf{type}$ using Σ and Id by

$$A(b) = \textstyle\sum_{b':B'} \mathsf{Id}_B(t(b'), b).$$

In words, $A(b)$ is the type of elements of B' that t sends to b. There is moreover an isomorphism of contexts θ given by the following mappings:

$$\begin{aligned} (b' : B') &\longrightarrow\!\!\!\!\cong\!\!\!\!\longrightarrow (b : B, c : A(b)) \\ b' &\longmapsto (t(b'), \langle b', \mathsf{refl}\rangle) \\ b' &\longleftarrow\!\shortmid\ (b, \langle b', p\rangle) \end{aligned}$$

Thus we have:

$$\begin{array}{ccc} 1.B' & \xrightarrow[\cong]{\theta} & 1.B.A \\ & {\scriptstyle\langle \mathsf{p}_p, t\rangle}\searrow \quad \swarrow{\scriptstyle \mathsf{p}_A} & \\ & 1.B & \end{array}$$

as desired. Note that the fact that these mappings cancel up to *judgmental* equality and therefore give an isomorphism of contexts depends crucially on equality reflection. □

Lemma 3.3 is a kind of *relative* democracy: it shows that the projection $\mathbf{Ty}_\Gamma \to \mathcal{C}/\Gamma$ from a type over Γ to a substitution into Γ is an equivalence.

Theorem 3.4. If $(\mathcal{C}, \pi)$ is a natural model with Σ and Id types, then $\mathcal{C}$ has finite limits.

Proof. It suffices to check that $\mathcal{C}$ has a terminal object and pullbacks. The former is true by definition of natural model. For the latter, let a diagram

$$\begin{array}{ccc} & & \Gamma' \\ & & \downarrow{\scriptstyle \gamma} \\ \Delta & \xrightarrow[\sigma]{} & \Gamma \end{array}$$

in $\mathcal{C}$ be given. Given that pullbacks are isomorphism invariant, it suffices to find a pullback for the isomorphic diagram

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

where A is obtained from Lemma 3.3. In this case, we can construct a pullback by applying the substitution σ to the type A. In the language of natural models, this means the following. The universal property of the pullback defining $\Gamma.A$ gives us a unique dashed map in the diagram

$$\begin{array}{ccccc} & & \overset{\mathsf{q}_{A\sigma}}{\frown} & & \\ \text{ѣ}(\Delta.A\sigma) & \dashrightarrow & \text{ѣ}(\Gamma.A) & \xrightarrow[\mathsf{q}_A]{} & \mathrm{Tm} \\ \downarrow{\scriptstyle \mathsf{p}_{A\sigma}} & & \downarrow{\scriptstyle \mathsf{p}_A} \lrcorner & & \downarrow{\scriptstyle \pi} \\ \text{ѣ}\Delta & \xrightarrow[\sigma]{} & \text{ѣ}\Gamma & \xrightarrow[A]{} & \mathrm{Ty}. \end{array}$$

Since both the right square and the outer rectangle are pullbacks, it follows from the pullback pasting lemma that the left square is also a pullback, and this is the pullback we were looking for. □

When $(\mathcal{C}, \pi)$ has Π types, we can use Lemma 3.3 in a similar fashion to prove that $\mathcal{C}$ is locally cartesian closed. Recall that a category $\mathcal{E}$ with finite limits is locally cartesian closed when all of its slice categories have exponentials. An equivalent condition is that for every $f\colon X' \to X$ in $\mathcal{E}$, the induced functor $f^*\colon \mathcal{E}/X \to \mathcal{E}/X'$ that pulls back a morphism along f has a right adjoint $f_*\colon \mathcal{E}/X' \to \mathcal{E}/X$ (called "pushforward").

In the case of $(\mathcal{C}, \pi)$, if we are given $\gamma\colon \Gamma' \to \Gamma$ and $\sigma\colon \Delta \to \Gamma'$ and want to define $\gamma_*\sigma$, we first replace the diagram

$$\begin{array}{ccc} & & \Delta \\ & & \downarrow{\scriptstyle \sigma} \\ \Gamma' & \xrightarrow[\gamma]{} & \Gamma \end{array}$$

by an isomorphic diagram

$$\begin{array}{ccc} \Gamma.A.B & & \\ \downarrow{\scriptstyle \mathsf{p}_B} & & \\ \Gamma.A & \xrightarrow[\mathsf{p}_A]{} & \Gamma \end{array}$$

by applying Lemma 3.3 twice; again, pushforwards are defined up to isomorphism, so it suffices to handle this case. We can show that the projection for the Π type

$$\begin{array}{ccc} \Gamma.A.B & & \Gamma.\Pi(A,B) \\ \downarrow{\scriptstyle \mathsf{p}_B} & & \vdots{\scriptstyle \mathsf{p}_{\Pi(A,B)}} \\ \Gamma.A & \xrightarrow[\mathsf{p}_A]{} & \Gamma \end{array}$$

has the universal property of $(\mathsf{p}_A)_*(\mathsf{p}_B)$.

In the next lecture, we will discuss the opposite direction: going from categories with finite limits to models of type theory with Σ and extensional Id.

References

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. "Explicit substitutions". In: *Journal of Functional Programming* 1.4 (1991), pp. 375–416. DOI: 10.1017/S0956796800000186.

[AG25] Carlo Angiuli and Daniel Gratzer. "Principles of Dependent Type Theory". Book draft. 2025. URL: https://www.danielgratzer.com/papers/type-theory-book.pdf.

[ALN24] Benedikt Ahrens, Peter LeFanu Lumsdaine, and Paige Randall North. "Comparing Semantic Frameworks for Dependently-Sorted Algebraic Theories". In: *Programming Languages and Systems*. Springer Nature Singapore, 2024, pp. 3–22. DOI: 10.1007/978-981-97-8943-6_1. arXiv: 2412.19946 [math.CT].

[Awo06] Steve Awodey. *Category Theory*. Oxford University Press, 2006. ISBN: 9780198568612. DOI: 10.1093/acprof:oso/9780198568612.001.0001.

[Awo16] Steve Awodey. "Natural models of homotopy type theory". In: *Mathematical Structures in Computer Science* 28.2 (2016), pp. 241–286. DOI: 10.1017/s0960129516000268.

[Boe20] Menno de Boer. *A Proof and Formalization of the Initiality Conjecture of Dependent Type Theory*. Licentiate. 2020. URL: https://su.diva-portal.org/smash/record.jsf?pid=diva2:1431287.

[Car78] John W. Cartmell. "Generalised Algebraic Theories and Contextual Categories". PhD thesis. Oxford University, 1978.

[CD14] Pierre Clairambault and Peter Dybjer. "The biequivalence of locally cartesian closed categories and Martin-Löf type theories". In: *Mathematical Structures in Computer Science* 24.6 (2014), e240606. DOI: 10.1017/S0960129513000881.

[CGH14] Pierre-Louis Curien, Richard Garner, and Martin Hofmann. "Revisiting the categorical interpretation of dependent type theory". In: *Theoretical Computer Science* 546 (2014), pp. 99–119. DOI: 10.1016/j.tcs.2014.03.003.

[Cur93] Pierre-Louis Curien. "Substitution up to Isomorphism". In: *Fundamenta Informaticae* 19.1–2 (1993), pp. 51–85. DOI: 10.3233/fi-1993-191-204.

[Dyb96] Peter Dybjer. "Internal type theory". In: *Types for Proofs and Programs: International Workshop TYPES '95, Torino, Italy*. Springer Berlin Heidelberg, 1996, pp. 120–134. DOI: 10.1007/3-540-61780-9_66.

[Hof95] Martin Hofmann. "On the interpretation of type theory in locally cartesian closed categories". In: *Computer Science Logic. 8th Workshop, CSL '94, Kazimierz, Poland, September 1994, Selected Papers*. Ed. by Leszek Pacholski and Jerzy Tiuryn. Springer Berlin Heidelberg, 1995, pp. 427–441. DOI: 10.1007/bfb0022273.

[Mar75] Per Martin-Löf. "An intuitionistic theory of types: predicative part". In: *Logic Colloquium '73*. Ed. by H.E. Rose and J.C. Shepherdson. Vol. 80. Studies in Logic and the Foundations of Mathematics. North-Holland, 1975, pp. 73–118. DOI: 10.1016/S0049-237X(08)71945-1.

[Mar82] Per Martin-Löf. "Constructive Mathematics and Computer Programming". In: *Logic, Methodology and Philosophy of Science VI*. Ed. by L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski. Vol. 104. Studies in Logic and the Foundations of Mathematics. Elsevier, 1982, pp. 153–175. DOI: 10.1016/S0049-237X(09)70189-2.

[Mar92] Per Martin-Löf. "Substitution calculus". Notes from a lecture given in Göteborg. Sept. 1992. URL: https://archive-pml.github.io/martin-lof/pdfs/Substitution-calculus-1992.pdf.

[MM94] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer New York, 1994. ISBN: 9781461209270. DOI: 10.1007/978-1-4612-0927-0.

[Pit01] Andrew M. Pitts. "Categorical Logic". In: *Handbook of Logic in Computer Science*. Vol. 5. Oxford University Press, 2001. ISBN: 9780191916663. DOI: 10.1093/oso/9780198537816.001.0001.

[See84] R. A. G. Seely. "Locally cartesian closed categories and type theory". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 95.1 (1984), pp. 33–48. DOI: 10.1017/s0305004100061284.

[Str91] Thomas Streicher. *Semantics of Type Theory. Correctness, Completeness and Independence Results*. Progress in Theoretical Computer Science. Birkhäuser Boston, 1991. DOI: 10.1007/978-1-4612-0433-6.

[Uem21] Taichi Uemura. "Abstract and Concrete Type Theories". PhD thesis. University of Amsterdam, 2021. URL: https://eprints.illc.uva.nl/id/eprint/2195/.