Notes on Type Theory

[DRAFT: April 8, 2025]

Steve Awodey

with contributions from Andrej Bauer

Contents

Chapter 4

Homotopy Type Theory

The extensional dependent type theory of the previous chapter is in some ways a very natural system that admits an intuitively clear model in the locally cartesian closed category of sets and related categories. But for computational purposes, and specifically for the important application of type theory to proof checking in a computer proof assistant such as Agda or Lean, it has some serious defects: the equality relation between terms (or types) is not *decidable*: there is no algorithm that will determine whether two closed terms of a given type $s, t : A$ are (judgementally) equal $s \equiv t : A$. Indeed, there is no normalization procedure for reducing terms to normal forms—otherwise we could use it to decide whether two terms were equal by normalizing them and then comparing their normal forms. Relatedly, one cannot effectively decide whether a given type (e.g an equality type such as $\mathtt{Eq}_A(s,t)$) is inhabited (which would be a decision procedure for the *provability* of $s =_A t$), even given a candidate "proof term" $p : \mathtt{Eq}_A(s,t)$ (which would be a decision procedure for *being a proof*).

For this reason, the extensional system is often replaced in applications by a weaker one, called *intensional type theory*, which enjoys better computational behavior, such as decidability of equality and type-checking, and normalization. A good discussion of these and several related issues, such as canonicity and consistency can be found in Chapter 3 of the book [AG].

However, this is only one side of the story. The intensional theory was mainly a technical device for specialists in computational type theory (and a conceptual challenge from the semantic point of view) until around 2006, when it was discovered that this theory admitted a homotopical (and higher-categorical) interpretation, which led to the discovery of Homotopy type theory (HoTT) [**?**]. This interpretation not only helped to clarify the intensional theory, and prove useful in investigating its computational properties, but also opened up a wide range of applications outside of the conventional areas of type theory, *vis.* computational and constructive mathematics. For, quite independently of such applications, the homotopical interpretation permits the use of intensional type theory as a powerful and expressive *internal language* for formal reasoning in homotopy theory and higher category theory, both highly abstract areas of mathematics, for which new and rigorous tools for calculation and proof are quite welcome. Moreover, the fortuitous fact that

this system also has the good computational behavior that it does has led to the use of computational proof assistants in homotopy theory and higher category theory, even ahead of some more down-to-earth branches of mathematics, where such exotic semantics were not needed.

The homotopical interpretation was already anticipated by a 2-dimensional one in the category of groupoids, a special case of a higher categorical model that already suffices to make some of the essential features of such models clear. Thus we shall briefly review this model below, after introducing the intensional theory, and before considering the general homotopical semantics using weak factorization systems. Such "weak" interpretations also bring to a head the coherence issues that we deferred in the previous chapter, and we conclude with one approach to strictifying such interpretations using natural models, aka, categories with families.

4.1 Identity types

We begin by recalling from Section **??** the rules for *equality* types in the *extensional* system: The formation, introduction, elimination, and computation rules for equality types were as follows:

$$\frac{s : A \qquad t : A}{s =_A t \ \ \mathtt{type}} \qquad\qquad \frac{a : A}{\mathtt{refl}(a) : (a =_A a)}$$

$$\frac{p : s =_A t}{s \equiv t : A} \qquad\qquad \frac{p : s =_A t}{p \equiv \mathtt{refl}(s) : (s =_A s)}$$

The *Identity types* in the intensional theory, also written $x =_A y$, or sometimes $\mathtt{Id}_A(x, y)$, have the same formation and introduction rules as the Equality types, but the *elimination rule* of "equality reflection" is replaced by the following elimination rule:

$$\frac{x : A, y : A, z : \mathtt{Id}_A(x, y) \vdash C(x, y, z) \ \mathtt{type}, \qquad x : A \vdash c(x) : C(x, x, \mathtt{refl}(x))}{x : A, y : A, z : \mathtt{Id}_A(x, y) \vdash \mathtt{J}(x, y, z, c) : C}$$

in which the variable x is bound in the occurance of c within the eliminator $\mathtt{J}$. The associated *computation rule* then becomes:

$$x : A \vdash \mathtt{J}(x, x, \mathtt{refl}(x), c) \equiv c(x) : C(x, x, \mathtt{refl}(x))$$

In HoTT, the elimination rule is called *path induction*, for reasons that will become clear.

To see how the elimination rule works, let us derive the basic laws of identity, namely reflexivity, symmetry, and transitivity, as well as Leibniz's Law the *indicernibility of identicals*, also known as the substitution of equals for equals.

- Reflexivity:

$$\frac{x : A, y : A, z : x =_A y \vdash x =_A y \ \ \mathtt{type}}{x : A \vdash \mathtt{refl}(x) : x =_A x}$$

- Symmetry:

$$\frac{x : A, y : A, z : x =_A y \vdash x =_A y \ \texttt{type}, \qquad x : A \vdash \texttt{refl}(x) : x =_A x}{x : A, y : A, z : x =_A y \vdash \texttt{J}(x, y, z, \texttt{refl}) : y =_A x}$$

- Transitivity: we wish to show

$$x : A, y : A, z : A, u : x =_A y, v : y =_A z \vdash ? : x =_A z$$

  regarding $z : A$ as a fixed parameter, in order to apply an `Id`-elim with respect to the assumption $u : x =_A y$ we set x to y, and seek a premiss of the form

$$y : A, z : A, v : y =_A z \vdash ? : y =_A z$$

  we cannot simply take v, since the order of the types in the context is still wrong for `Id`-elim, but we can (reorder y, z and), project from the assumption $v : y =_A z$ and then move it to the right with a λ-abstraction to obtain

$$z : A, y : A \vdash \ \lambda v \,.\, v : y =_A z \to y =_A z \,,$$

  and now we can apply the planned `Id`-elim with respect to $u : x =_A y$ to obtain

$$z : A, y : A, x : A, u : x =_A y \vdash \texttt{J}(x, y, u, \lambda v.v) : y =_A z \to x =_A z$$

  from which follows the desired

$$x : A, y : A, z : A, u : x =_A y, v : y =_A z \vdash \texttt{J}(x, y, u, \lambda v.v)\, v : x =_A z \,.$$

- Substitution:

$$\frac{x : A \vdash C(x) \ \texttt{type}}{x : A, y : A, u : x =_A y \vdash ? : C(x) \to C(y)}$$

  it suffices to have

$$x : A \vdash c(x) : C(x) \to C(x)$$

  for this, we can take $c(x) = \lambda z : C(x).\, z : C(x) \to C(x)$ to obtain

$$x : A, y : A, u : x =_A y \vdash \texttt{J}(x, y, u, x.\lambda z : C(x).z) : C(x) \to C(y) \,.$$

  Note that the variable x is bound in the `J` term.

Many more properties of `Id`-types and their associated `J`-terms are shown in the introductory texts [Uni13, Rij25]. One key fact is that the higher identity types $\texttt{Id}_{\texttt{Id}_A(a,b)}(p, q)$ are no longer degenerate, but themselves may have terms that are non-identical, i.e. not propositionally equal, leading to so-called *higher types*. This "failure of UIP" (uniqueness of identity proofs) in the intensional system was first shown using the groupoid model, which sheds considerable light on the intensional system.

Exercise 4.1.1. Show that given any $a, b, c : A$ and $p : a =_A b$ and $q : b =_A c$, one can define a composite $p \cdot q : a =_A c$ (using the transitivity of $=_A$). Then show that, for any $p : a =_A b$, the symmetry term $\sigma(p) : b =_A a$ satisfies the (propositional) equation $\sigma(p) \cdot p = \mathtt{refl}$. Is either of $\sigma(p) \cdot p = \mathtt{refl}$ or $\mathtt{refl} \cdot \sigma(p) = \mathtt{refl}$ judgemental? What about associativity of $p \cdot q$

Exercise 4.1.2. Show that $p \cdot q$ from the previous exercise is (propositionally) associative.

Exercise 4.1.3. Show that any term $f : A \to B$ acts on identities $p : a =_A b$, in the sense that there is a term $\mathtt{ap}(f)(p) : fa =_B fb$. Is $\mathtt{ap}(f)$ "functorial" (in the evident sense)?

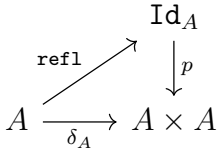
Exercise 4.1.4. Observe that the Substitution property means that the assignment

$$(a : A) \mapsto C(a) \ \mathtt{type}$$

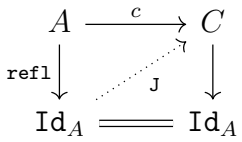
is functorial (in some sense). Is it strictly functorial?

4.1.1 The naive interpretation

We can try to interpret the (intensional) identity types in the naive way, as we did for extensional dependent type theory. This would give the formation and introduction rules as a type family $\mathtt{Id}_A \to A \times A$ with a partial section over the diagonal substitution $\delta_A : (x : A) \to (x : A, y : A)$.



where we are writing $\mathtt{Id}_A$ for the extended context $(A, A, \mathtt{Id}_A)$ and p for the dependent family $(x : A, y : A \vdash \mathtt{Id}_A(x, y))$. The elimination rule then takes the form:

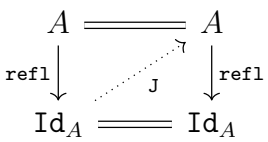


for any type family $C \to \mathtt{Id}_A$, with the computation rule asserting that the top triangle commutes (the bottom triangle commutes by the assumtion tht $\mathtt{J}$ is a section of $C \to \mathtt{Id}_A$).

But now recall that in extensional type theory, *any* map $f : B \to A$ can be regarded as a type family over A, namely by taking the *graph factorization*

$$B \cong \Sigma_{a:A}\Sigma_{b:B}\mathtt{Eq}_A(a, fb) \longrightarrow B \times A.$$

So we can take the family C in the elimination to be $\mathtt{refl} : A \to \mathtt{Id}_A$, to obtain:



We therefore get an iso $A \cong \mathtt{Id}_A$, making the identity type isomorphic to the extensional equality type $\mathtt{Eq}_A$.

Exercise 4.1.5. Prove that in the extensional theory, the graph factorization does indeed make any map $f : B \to A$ isomorphic to a family of types over its codomain.

4.2 The groupoid model

The groupoid model violates UIP.

4.3 Weak factorization systems

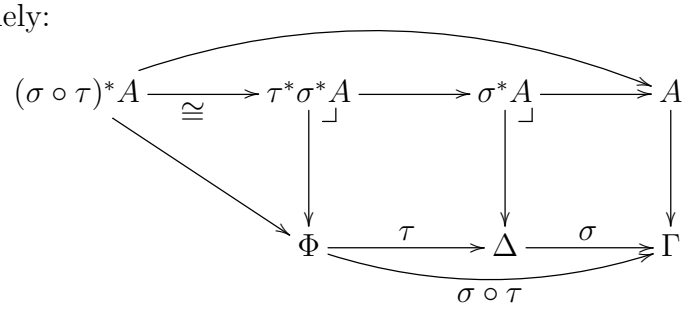
WFSs generalize the groupoid model and include many familiar examples. But strictification is a more serious problem because of the J-term.

4.4 Natural models

The semantics of DTT in LCCCs described in the previous sections uses the "slice category" hyperdoctrine of an LCCC to interpret the dependent types. Thus the contexts Γ and substitutions $\sigma : \Delta \to \Gamma$ are interpreted as the objects and arrow of a LCC category $\mathcal{C}$, and the dependent types $\Gamma \vdash A$ and terms $\Gamma \vdash a : A$ are interpreted as objects $A \to \Gamma$ in the slice category $\mathcal{C}/\Gamma$ and their global sections $a : \Gamma \to A$ (over Γ). As we mentioned in Remark **??**, however, there is a problem with this kind of semantics (as first pointed out by [Hof95]): as a hyperdoctrine, this interpretation is a *pseudo*functor $\mathcal{C}/ : \mathcal{C}^{\mathsf{op}} \to \mathsf{Cat}$, but the syntax of DTT produces an actual *presheaf* of types in context $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$, since substitution into dependent types is *strictly* functorial with respect to composition of substitutions, in the sense that for a type in context $\Gamma \vdash A$ and substitutions $\sigma : \Delta \to \Gamma$ and $\tau : \Phi \to \Delta$ we have an *equality of types in context*,

$$\Phi \vdash (A[\sigma])[\tau] \equiv A[\sigma \circ \tau]\,,$$

rather than the (canonical) isomorphism $\cong$ fitting into the two-pullbacks diagram of the hyperdoctrine, namely:



A similar problem occurs in the Beck-Chavalley conditions, where the hyperdoctrine structure has only canonical isos, rather than the strict equalities that obtain in the syntax, such as

$$(\Pi_{x:A} B)[\sigma] \equiv (\Pi_{x:A[\sigma]} B[\sigma])\,.$$

There are various different solutions to this problem in the literature, some involving "strictifications" of the LCC slice-category hyperdoctrine (including both left- and right-adjoint strictifications), as well as other semantics altogether, such as categories-with-families [Dyb96], categories-with-attributes [**?**], and comprehension categories [**?**].

A solution based on the notion of *universe* $\widetilde{U} \to U$ was first proposed by Voevodsky; this approach is combined with the notion of a *representable natural transformation* in [Awo16] as follows.

Definition 4.4.1. For a small category $\mathbb{C}$, a natural transformation $f : Y \to X$ of presheaves on $\mathbb{C}$ is called *representable* if for every $C \in \mathbb{C}$ and $x \in X(C)$, there is given a $D \in \mathbb{C}$, a $p : D \to C$, and a $y \in Y(D)$ such that the following square is a pullback.

$$\begin{array}{ccc} \mathsf{y}D & \xrightarrow{y} & Y \\ {\scriptstyle \mathsf{y}p}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathsf{y}C & \xrightarrow[x]{} & X \end{array} \tag{4.1}$$

We will show that a representable natural transformation is essentially the same thing as a *category with families* in the sense of Dybjer [Dyb96]. Indeed, let us write the objects of $\mathbb{C}$ as $\Gamma, \Delta, \ldots$ and the arrows as $\sigma : \Delta \to \Gamma, \ldots$, thinking of $\mathbb{C}$ as a "category of contexts". Let $\mathsf{u} : \dot{\mathsf{U}} \to \mathsf{U}$ be a representable map of presheaves, and write its elements as:

$$\begin{aligned} A \in \mathsf{U}(\Gamma) \quad &\text{iff} \quad \Gamma \vdash A \\ a \in \dot{\mathsf{U}}(\Gamma) \quad &\text{iff} \quad \Gamma \vdash a : A, \end{aligned}$$

where $A = \mathsf{u} \circ a$, as indicated in:

$$\begin{array}{ccc} & & \dot{\mathsf{U}} \\ & {\scriptstyle a}\nearrow & \downarrow{\scriptstyle \mathsf{u}} \\ \mathsf{y}\Gamma & \xrightarrow[A]{} & \mathsf{U} \end{array}$$

Thus we regard U as the *presheaf of types*, with $\mathsf{U}(\Gamma)$ the set of all types in context Γ, and $\dot{\mathsf{U}}$ as the *presheaf of terms*, with $\dot{\mathsf{U}}(\Gamma)$ the set of all terms in context Γ, while the component $\mathsf{u}_\Gamma : \dot{\mathsf{U}}(\Gamma) \to \mathsf{U}(\Gamma)$ is the typing of the terms in context Γ.

The naturality of $\mathsf{u} : \dot{\mathsf{U}} \to \mathsf{U}$ just means that for any substitution $\sigma : \Delta \to \Gamma$, we have an action on types and terms:

$$\begin{aligned} \Gamma \vdash A \quad &\mapsto \quad \Delta \vdash A\sigma \\ \Gamma \vdash a : A \quad &\mapsto \quad \Delta \vdash a\sigma : A\sigma\,. \end{aligned}$$

While, by functoriality, given any further $\tau : \Phi \to \Delta$, we have

$$(A\sigma)\tau = A(\sigma \circ \tau) \qquad (a\sigma)\tau = a(\sigma \circ \tau),$$

as well as

$$A1 = A \qquad a1 = a$$

for the identity substitution $1 : \Gamma \to \Gamma$.

Finally, the representability of the natural transformation $p : E \to U$ is exactly the operation of *context extension*: given any $\Gamma \vdash A$, by Yoneda we have the corresponding map $A : \mathsf{y}\Gamma \to \mathsf{U}$, and we let $p_A : \Gamma.A \to \Gamma$ be (the map representing) the pullback of u along A, as in (4.1). We therefore have a pullback square:

$$\begin{array}{ccc} \mathsf{y}\Gamma.A & \xrightarrow{q_A} & \dot{\mathsf{U}} \\ {\scriptstyle \mathsf{y}p_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{u}} \\ \mathsf{y}\Gamma & \xrightarrow[A]{} & \mathsf{U} \end{array} \tag{4.2}$$

where the map $q_A : \Gamma.A \to \dot{\mathsf{U}}$ now determines a term

$$\Gamma.A \vdash q_A : Ap_A.$$

We may hereafter omit the y for the Yoneda embedding, letting the Greek letters serve to distinguish representable presheaves and their maps.

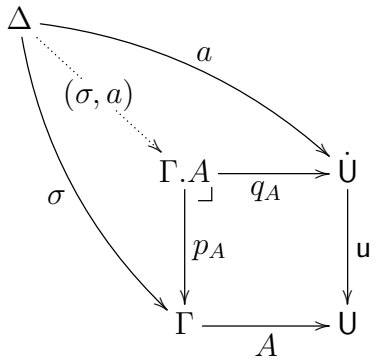
Exercise 4.4.2. Show that the fact that (4.2) is a pullback means that given any $\sigma : \Delta \to \Gamma$ and $\Delta \vdash a : A\sigma$, there is a map

$$(\sigma, a) : \Delta \to \Gamma.A,$$

and this operation satisfies the equations

$$p_A \circ (\sigma, a) = \sigma$$
$$q_A(\sigma, a) = a,$$

as indicated in the following diagram.



Show moreover that the uniqueness of (σ, a) means that for any $\tau : \Delta' \to \Delta$ we also have:

$$(\sigma, a) \circ \tau = (\sigma \circ \tau, a\tau)$$
$$(p_A, q_A) = 1.$$

Comparing the foregoing with the definition of a category with families in [Dyb96], we have shown:

Proposition 4.4.3. *Let* $\mathsf{u} : \dot{\mathsf{U}} \to \mathsf{U}$ *be a representable natural transformation of presheaves on a small category* $\mathbb{C}$ *with a terminal object. Then* u *determines a category with families, with* $\mathbb{C}$ *as the contexts and substitutions,* $\mathsf{U}(\Gamma)$ *as the types in context* Γ*, and* $\dot{\mathsf{U}}(\Gamma)$ *as the terms in context* Γ*.*

Remark 4.4.4. A category with families is usually defined in terms of a presheaf

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

of types on the category $\mathcal{C}$ of contexts, together with a presheaf

$$\mathsf{Tm}' : (\textstyle\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$$

of typed-terms on the category $\int_{\mathcal{C}} \mathsf{Ty}$ of types-in-context. We are using the equivalence of categories, valid for any category of presheaves $\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}$,

$$\mathsf{Set}^{\mathbb{C}^{\mathsf{op}}}\!/_P \;\simeq\; \mathsf{Set}^{(\int_{\mathbb{C}} P)^{\mathsf{op}}}$$

between the slice category over a presheaf P and the presheaves on its category of elements $\int_{\mathbb{C}} P$, to turn the presheaf $\mathsf{Tm}' : (\int_{\mathcal{C}} \mathsf{Ty})^{\mathsf{op}} \to \mathsf{Set}$ into one $\mathsf{Tm} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$ together with a map $\mathsf{Tm} \to \mathsf{Ty}$ in $\mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$.

We think of a representable map of presheaves on an arbitrary category $\mathbb{C}$ as a "type theory over $\mathbb{C}$", with $\mathbb{C}$ as the category of contexts and substitutions. We will show in Section 4.4.2 that such a map of presheaves is essentially determined by a class of maps in $\mathbb{C}$ that is closed under all pullbacks, corresponding to the "incoherent" interpretation of types in context as maps $A \to \Gamma$.

Definition 4.4.5. A *natural model of type theory* on a small category $\mathbb{C}$ is a representable map of presheaves $\mathsf{u} : \dot{\mathsf{U}} \to \mathsf{U}$.

Exercise 4.4.6 (The natural model of syntax)**.** Let $\mathbb{T}$ be a dependent type theory and $\mathcal{C}_{\mathbb{T}}$ its category of contexts and substitutions. Define the presheaves $\mathsf{Ty} : {\mathcal{C}_{\mathbb{T}}}^{\mathsf{op}} \to \mathsf{Set}$ of types-in-context and $\mathsf{Tm} : {\mathcal{C}_{\mathbb{T}}}^{\mathsf{op}} \to \mathsf{Set}$ of terms-in-context, along with a natural transformation

$$\mathsf{tp} : \mathsf{Tm} \to \mathsf{Ty}$$

that takes a term to its type. Show that $\mathsf{tp} : \mathsf{Tm} \to \mathsf{Ty}$ is a natural model of type theory.

4.4.1 Modeling the type formers

4.4.2 Strictification

4.5 Universes

Bibliography

[AG] C. Angiuli and D. Gratzer. Principles of dependent type theory. Online at `https://carloangiuli.com/courses/b619-sp24/notes.pdf`. Version 2024-11-26.

[Awo16] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28:1–46, 11 2016.

[Dyb96] P. Dybjer. Internal type theory. *LNCS*, 1158:120–134, 1996.

[Hof95] Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and logics of computation*, volume 14 of *Publ. Newton Inst.*, pages 79–130. Cambridge University Press, Cambridge, 1995.

[Rij25] Egbert Rijke. *Introduction to Homotopy Type Theory*. Cambridge University Press, 2025.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.