Dependent Type Theories à la Carte

Mitchell Riley

13th Sep 2022

Dependent type theories are formal systems for working internal to (higher) categories.

Dependent Type Theory

Ordinary Foundations:

- First order logic ($\forall$, $\exists$, $\top$, $\bot$, $\wedge$, $\vee$, $=$)
- Membership relation ($\in$)
- ZFC axioms
- Possibly more axioms

Dependent Type Theory:

- Membership is 'built in'
- Pair types $\times$
- Function types $\rightarrow$
- Identity types $a = a'$
- Universe type $\mathcal{U}$
- Possibly some inductive types $(0, \mathbb{N}, \dots)$
- Possibly some axioms

Dependent Type Theory

In First Order Logic, there are two main *judgements*:

- $x_1, \ldots, x_n \vdash \psi$ prop
- $x_1, \ldots, x_n \mid \phi_1, \ldots, \phi_n \vdash \psi$ true

One might ask whether $x \in z$ true.

In Dependent Type Theory, there are also two:

- $x_1 : A_1, \ldots, x_n : A_n \vdash B$ type
- $x_1 : A_1, \ldots, x_n : A_n \vdash b : B$

Every term comes with its type.

Dependent Type Theory

Working in Dependent Type Theory feels a lot like working with ordinary sets.

Pairs and functions are *primitive*, rather than being constructed out of sets.

$$f : A \times B \to A \times (B \times A)$$
$$f(x, y) := (x, (y, x))$$

Dependent Type Theory

$$x : \mathrm{Month} \vdash \mathrm{DayOf}(x) \ \mathsf{type}$$
$$x : M \vdash T_x M \ \mathsf{type}$$
$$R : \mathrm{Ring} \vdash \mathrm{Mod}(R) \ \mathsf{type}$$
$$X : \mathrm{Top}, c : \mathrm{Cover}(X) \vdash \mathrm{Subcover}(X, c) \ \mathsf{type}$$

It is natural to consider *dependent* pairs:

Example

$(x : \mathrm{Month}) \times \mathrm{DayOf}(x)$ is type of all days in the year.

$(x : M) \times T_x M$ is the tangent bundle TM.

Rules for Dependent Pairs

- Given A and $B(x)$ that may depend on $x : A$, there is a type

$$(x : A) \times B(x)\ \mathsf{type}$$

- For any $a : A$ and $b : B(a)$, we can form the pair

$$(a, b) : (x : A) \times B(x)$$

- For any $p : (x : A) \times B(x)$, we can take the first and second projection

$$\mathsf{pr}_1(p) : A$$
$$\mathsf{pr}_2(p) : B(\mathsf{pr}_1(p))$$

Judgements and Rules

$$\textsc{rule-name}\ \frac{\mathcal{J}_1 \qquad \dots \qquad \mathcal{J}_n \qquad \text{(premises)}}{\mathcal{J} \qquad \text{(conclusion)}}$$

$$\textsc{var}\ \frac{}{\Gamma, x : A, \Gamma' \vdash x : A}$$

$$\times\text{-}\textsc{form}\ \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B\ \mathsf{type}}{\Gamma \vdash (x : A) \times B\ \mathsf{type}}$$

$$\times\text{-}\textsc{intro}\ \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : (x : A) \times B}$$

$$\times\text{-}\mathsf{pr}_1\ \frac{\Gamma \vdash p : (x : A) \times B}{\Gamma \vdash \mathsf{pr}_1(p) : A} \qquad \times\text{-}\mathsf{pr}_2\ \frac{\Gamma \vdash p : (x : A) \times B}{\Gamma \vdash \mathsf{pr}_2(p) : B[\mathsf{pr}_1(p)/x]}$$

Identity Types

- Form: For any A and elements $a : A$, $a' : A$, there is a type of *identifications* of a with a', called $a =_A a'$.
- Intro: There is an identification from any $a : A$ to itself called $\mathsf{refl}_a : a =_A a$.
- Elim: To prove anything using $a =_A a'$, it suffices to prove it for a generic $\mathsf{refl}_w : w =_A w$.

$$=\text{-FORM}\ \frac{\Gamma \vdash a : A \qquad \Gamma \vdash a' : A}{\Gamma \vdash a =_A a' \ \mathsf{type}} \qquad =\text{-INTRO}\ \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}$$

$$=\text{-ELIM}\ \frac{\begin{array}{c}\Gamma, x : A, y : A, z : x =_A y \vdash C \ \mathsf{type} \\ \Gamma, w : A \vdash c : C[w/x, w/y, \mathsf{refl}_w/z] \\ \Gamma \vdash p : a =_A a'\end{array}}{\Gamma \vdash \mathsf{let}\ \mathsf{refl}_w := p\ \mathsf{in}\ c : C[a/x, b/y, p/z]}$$

Identity Types

For some types, $a = a'$ does behave just like ordinary equality.
The *statement* of commutativity of addition is the type

$$(n : \mathbb{N}) \to (m : \mathbb{N}) \to (n + m = m + n)$$

A *proof* of commutativity is a function of this type.

Definition

A type A is *contractible* if there is a term of the type

$$\text{isContr}(A) := (c : A) \times ((x : A) \to (c = x))$$

(Don't worry, this doesn't mean just path-connected!)

Definition

The *fiber* of a function $f : A \to B$ over a point $b : B$ is

$$\text{fib}_f(b) := (x : A) \times (f(x) = b)$$

Definition

A function is an *equivalence* if the fiber over every point is contractible:

$$\text{isEquiv}(f) := (b : B) \to \text{isContr}(\text{fib}_f(b))$$

Interpretation into Categories

Γ ctx	Object Γ
$\Gamma \vdash A$ type	$A \to \Gamma$ in $\mathcal{C}/\Gamma$
$(x : A) \times B$	$\Sigma_A : \mathcal{C}/A \to \mathcal{C}/\Gamma$ on B
$(x : A) \to B$	$\Pi_A : \mathcal{C}/A \to \mathcal{C}/\Gamma$ on B
$x_1 = x_2$	Path space $PA \to A \times_\Gamma A$ in $\mathcal{C}/A \times_\Gamma A$
$\ldots$	$\ldots$

Theorem (Shulman 2019)

Every ∞-topos can be presented by a model category that admits a model of HoTT. (modulo closure of universes under HITs)

Homotopy Type Theory

With a few more type formers (some higher inductive types, univalent universes) the system is called Homotopy Type Theory.

Theorem (Licata, Shulman)

Let S^1 be the type freely generated by the terms $\mathsf{base} : S^1$ *and* $\mathsf{loop} : \mathsf{base} =_{S^1} \mathsf{base}$. *Then* $(\mathsf{base} =_{S^1} \mathsf{base}) \simeq \mathbb{Z}$.

Some other synthetic results:

- Some homotopy groups of spheres (Shulman, Brunerie, Licata)
- Freudenthal Suspension Theorem (Lumsdaine, Licata)
- Localisation (Christensen, Opie, Rijke, Scoccola)
- Blakers–Massey Theorem (Anel, Biedermann, Finster, Joyal)
- Serre Spectral Sequence (Avigad, Awodey, Buchholtz, Rijke, Shulman, van Doorn)

Cohesive Type Theory

A cohesive topos $\mathcal{H}$ is one equipped with an adjoint quadruple

$$\begin{array}{c} \mathcal{H} \\ \downarrow \Pi_0 \quad \uparrow \text{disc} \quad \downarrow \Gamma \quad \uparrow \text{codisc} \\ \mathcal{S} \end{array}$$

(+ some conditions)

Examples

- $\mathrm{Sh}(\mathrm{CartSp}_{\mathrm{top}})$: Topological homotopy types
- $\mathrm{Sh}(\mathrm{CartSp}_{\mathrm{smooth}})$: Smooth homotopy types
- $\mathrm{PSh}(\mathrm{Glo})$: Global equivariant homotopy types
- $\mathrm{PSh}(\Delta)$: Simplicial homotopy types

Cohesive Type Theory

We want to use these adjoints in type theory.

- $\flat :\equiv \mathrm{disc} \circ \Gamma$ (retopologise discretely)
- $\sharp :\equiv \mathrm{codisc} \circ \Gamma$ (retopologise codiscretely)

Theorem (Shulman)

Any internal coreflector on Type *has the form* $\Box A \simeq A \times U$ *for some proposition* U*.*

The problem is that the universal property applies in any context. I.e., that, if B is in the coreflective subcategory,

$$(\epsilon_A \circ -) : (B \to \Box A) \to (B \to A)$$

is an equivalence.

Cohesive Type Theory

Following the pattern of adjoint logic, we put in a judgemental version of $\flat$ and have the type formers interact with it.

$$\Delta \mid \Gamma \vdash a : A \qquad \text{corresponds to} \qquad a : \flat\Delta \times \Gamma \to A$$

We need two variable rules:

$$\textsc{var}\ \frac{}{\Delta \mid \Gamma, x : A, \Gamma' \vdash x : A} \qquad \textsc{var-crisp}\ \frac{}{\Delta, x :: A, \Delta' \mid \Gamma \vdash x : A}$$

The second rule comes from the counit $\flat A \to A$.

Cohesive Type Theory

How to think about the different kinds of assumptions?

- $\Delta \mid \Gamma, x : A, \Gamma' \vdash b : B$ means b varies continuously over A.
- $\Delta, x :: A, \Delta' \mid \Gamma \vdash b : B$ means B varies (possibly) discontinuously over A.

The introduction rule for $\flat$ is restricted:

$$\flat\text{-INTRO}\ \frac{\Delta \mid \cdot \vdash a : A}{\Delta \mid \Gamma \vdash a^{\flat} : \flat A}$$

This rescues us from the no-go theorem: we can only show

$$\flat(B \to \flat A) \to \flat(B \to A)$$

is an equivalence.

Commuting Cohesions?

TED-K involves multiple notions of cohesion. How can we use all of them in a single type theory?

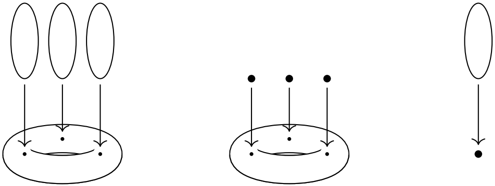
Parameterised Spectra

"Definition"

A *spectrum* is an object that represents a cohomology theory.

"Definition"

A *parameterised spectrum* is a bundle of spectra over a space.



Theorem (Biedermann, Joyal 2008)

The ∞-category of parameterised spectra $P\mathrm{Spec}$ is an ∞-topos.

Almost Cohesive Type Theory

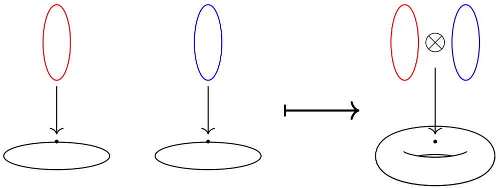
Comparing the setting of Cohesive Type Theory:

$$
\begin{array}{c}
\mathcal{E} \\
\Pi_0 \Big\downarrow \ \text{disc} \Big\uparrow \ \Big\downarrow \Gamma \ \Big\uparrow \text{codisc} \\
\mathcal{S}
\end{array}
\qquad\qquad
\begin{array}{c}
P\text{Spec} \\
0 \Big\uparrow \dashv \Big\downarrow \dashv \Big\uparrow 0 \\
\mathcal{S}
\end{array}
$$

We could use Cohesive Type Theory by asserting $\flat A \to A \to \sharp A$ is an equivalence.

Smash Product

For two types A and B, there should be a type $A \otimes B$ that corresponding to the 'external smash product'.



Linear Homotopy Type Theory

- (Vákár 2014) has linear type formers, but its dependent pairs/functions work differently to MLTT
- (Isaev 2021; Krishnaswami, Pradic, and Benton 2015) are 'LNL' type theories that separate linear types from non-linear types, so existing synthetic results can't be used
- (McBride 2016; Atkey 2018) are 'quantitative type theories' with only one kind of type, but do not allow 'ordinary' dependence

These mostly have models in monoidal fibrations $\mathcal{L} \to \mathcal{C}$, where $\mathcal{C}$ is a topos.

In our setting we can do better: $P\mathrm{Spec} \to \mathcal{S}$ is a monoidal fibration *and* $P\mathrm{Spec}$ *is a topos*.

Theorem

The universe of types is equivalent to

$$\mathcal{U} \simeq (X : \mathrm{Space}) \times (E : X \to \natural\mathrm{Spec}) \times ((x : X) \to \Sigma(E(x))_\natural)$$

Can this type theory formalise any of the work in the *Differential Cohomology* and *Proper Orbifold Cohomology* papers?

Robert Atkey (2018). "Syntax and Semantics of Quantitative Type Theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. DOI: 10.1145/3209108.3209189.

Valery Isaev (2021). "Indexed type theories". In: *Mathematical Structures in Computer Science* 31.1. DOI: 10.1017/S0960129520000092.

André Joyal (2008). *Notes on Logoi*. URL: http://www.math.uchicago.edu/~may/IMA/JOYAL/Joyal.pdf.

Neelakantan R. Krishnaswami, Pierre Pradic, and Nick Benton (2015). "Integrating Linear and Dependent Types". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. DOI: 10.1145/2676726.2676969.

Conor McBride (2016). "I Got Plenty o' Nuttin'". In: *A list of successes that can change the world*. Vol. 9600. DOI: 10.1007/978-3-319-30936-1_12.

References II

Michael Shulman (2019). *All $(\infty, 1)$-toposes have strict univalent universes.* arXiv: 1904.07004 [math.AT].

Matthjis Vákár (2014). *Syntax and Semantics of Linear Dependent Types.* arXiv: 1405.0033 [cs.AT].