Using Linear Homotopy Type Theory Informally

Mitchell Riley
Dan Licata
Wesleyan University

Following jww. Eric Finster
University of Cambridge

16th October 2021

Intended Models

Space-parameterised families of Spectra

Or more generally:

$\mathcal{X}$-parameterised families of $\mathcal{C}$

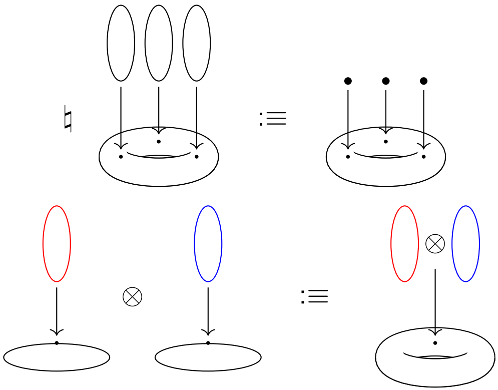
where

- $\mathcal{X}$ is an ∞-topos,
- $\mathcal{C}$ is a symmetric monoidal closed ∞-category *with a zero object*.

($\mathcal{C}$ for which $\mathcal{X}$-parameterised families form an ∞-topos are called an '∞-locus', Hoyois 2019)

Every object has a nonlinear aspect and a linear aspect.

Intended Models



- $\natural$: Extracts the nonlinear aspect of a type,
  - jww. Eric Finster, [arXiv: 2102.04099]
- $\otimes$: 'Fibrewise' tensor product,
- $\mathbb{S}$: Unit of $\otimes$,
- $\multimap$: Right adjoint to $\otimes$.

Eg. (Co)homology

The *homology and cohomology of X with coefficients in E* can be defined by

$$E_n(X) :\equiv \pi_n^s(\Sigma^\infty(X) \otimes E)$$
$$E^n(X) :\equiv \pi_n^s(\Sigma^\infty(X) \multimap E)$$

where

$$\pi_n^s(E) :\equiv \natural(\mathbb{S} \to E)$$
$$\Sigma^\infty(X) :\equiv X \wedge \mathbb{S}$$

New Type Formers

HoTT does not have type formers for these. So let's add them.
We want the output of the type formers to be *ordinary types*.

Cannot use an indexed type theory (Vákár 2014; Krishnaswami, Pradic, and Benton 2015; Isaev 2021), or quantitative type theory (McBride 2016; Atkey 2018; Moon, Eades III, and Orchard 2021; Fu, Kishida, and Selinger 2020)

♮

Marked Variable Uses

An extra variable rule, meaning only the non-linear aspect of an assumption is used.

- For any assumption $x : A$ there is a term $\underline{x} : \text{markFV}(A)$ where markFV(a) marks all uses of free variables in a.

$$\text{markFV}(x) \equiv \underline{x}$$
$$\text{markFV}(\lambda y.y + x) \equiv \lambda y.y + \underline{x}$$

- Substitution into $\underline{x}$ is defined by $\underline{x}[a/x] :\equiv \text{markFV}(a)$.

Definition

A term a is *dull* if $\text{markFV}(a) \equiv a$

So a only uses the non-linear aspect of the context.

Write the markFV(a) operation also as $\underline{a}$, so $\underline{x}[a/x] :\equiv \underline{a}$.

Rules for $\natural$

- Formation: For any dull $\underline{A} : \mathcal{U}$, there is a type $\natural\underline{A} : \mathcal{U}$.
- Introduction: For any dull term $\underline{a} : \underline{A}$, there is a term $\underline{a}^{\natural} : \natural\underline{A}$.
- Elimination: For any term $n : \natural\underline{A}$, there is a term $n_{\natural} : \underline{A}$.
- Computation: $\underline{a}^{\natural}{}_{\natural} \equiv \underline{a}$ for any $\underline{a} : \underline{A}$.
- Uniqueness: $n \equiv \underline{n_{\natural}}^{\natural}$ for any $n : \natural\underline{A}$.

(In the formalism this is described using a 'modal' context extension rather than a dullness side condition.)

The Symmetry Proof We Want

Proposition

sym : $A \otimes B \simeq B \otimes A$

Proof.

To define sym : $A \otimes B \to B \otimes A$, suppose we have $p : A \otimes B$. Then $\otimes$-induction allows us to assume $p \equiv x \otimes y$, and we have $y \otimes x$.

$$\mathsf{sym} :\equiv \lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ y \otimes x$$

Then to prove $\prod_{(p:A\otimes B)} \mathsf{sym}(\mathsf{sym}(p)) = p$, use $\otimes$-induction again: the goal reduces to $x \otimes y = x \otimes y$ for which we have reflexivity.

$$\mathsf{inv} :\equiv \lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ \mathsf{refl}_{x \otimes y}$$

□

Colourful Variables

We need to prevent terms like $\lambda x.x \otimes x : A \to A \otimes A$, so variable use needs to be restricted somehow.

- Every variable x has a *colour* $\mathfrak{c}$.
- The relationships between colours are collected in a *palette*.

Palettes Φ are constructed by

$$1 \qquad \Phi_1 \otimes \Phi_2 \qquad \Phi_1, \Phi_2 \qquad \mathfrak{c} \qquad \mathfrak{c} \prec \Phi$$

Typical palettes:

$$\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b} \qquad \mathfrak{w} \prec (\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}) \otimes \mathfrak{y} \qquad \mathfrak{p} \prec (\mathfrak{r} \otimes \mathfrak{b}, \mathfrak{r}' \otimes \mathfrak{b}')$$

(Similar to ‘bunched’ type theory P. W. O’Hearn and Pym 1999; P. O’Hearn 2003)

Using Colourful Variables

We need to keep track of the current ‘top colour’. Suppose $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$, and we have variables $x^{\mathfrak{r}} : A$, $y^{\mathfrak{b}} : B$, $z^{\mathfrak{p}} : C$.

- To be well-formed, a term must ‘be purple’.
- Only $z : C$ is a well-formed term using the normal variable rule.
- Each of $\underline{x} : \underline{A}, \underline{y} : \underline{B}, \underline{z} : \underline{C}$ is well-formed: any variable can be used marked.

Ordinary type formers bind variables with the current top colour:

$$\sum_{(x:A)} B(x) \qquad \prod_{(x:A)} B(x) \qquad (\lambda x.b)$$

$$\mathsf{ind}_{+}(z.C, x.c_1, y.c_2, p) \qquad \mathsf{ind}_{=}(x.x'.p.C, x.c, p)$$

Rules for $\otimes$

Let $\mathfrak{p}$ be the top colour.

- Formation: If $\underline{A} : \mathcal{U}$ and $\underline{B} : \mathcal{U}$, then $\underline{A} \otimes \underline{B} : \mathcal{U}$.
- Introduction: For any* $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$ and terms $a : \underline{A}$ with colour $\mathfrak{r}$ and $b : \underline{B}(\underline{a})$ with colour $\mathfrak{b}$, there is a term

$$a \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; b : \bigodot_{(\underline{x}:\underline{A})} \underline{B}(\underline{x})$$

- Elimination: Any term $p : \bigodot_{(\underline{x}:\underline{A})} \underline{B}(\underline{x})$ may be assumed to be of the form $x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y$ for some variables $x^{\mathfrak{r}} : \underline{A}, y^{\mathfrak{b}} : \underline{B}(\underline{x})$ with $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$ in a term $c : C[x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y/z]$.

$$(\text{let } x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y = p \text{ in } c) : C[p/z]$$

- Computation: If the term really is of the form $a \;{}_{\mathfrak{r}'}\!\otimes_{\mathfrak{b}'}\; b$, then

$$(\text{let } x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y = a \;{}_{\mathfrak{r}'}\!\otimes_{\mathfrak{b}'}\; b \text{ in } c) \equiv c[\mathfrak{r}'/\mathfrak{r} \otimes \mathfrak{b}'/\mathfrak{b} \mid a/x, b/x]$$

Eg: Symmetry

Proposition

There is a function $\mathsf{sym} : \underline{A} \otimes \underline{B} \to \underline{B} \otimes \underline{A}$

Proof.

Suppose have $p : \underline{A} \otimes \underline{B}$. Then $\otimes$-induction on p gives $x^{\mathfrak{r}} : \underline{A}$ and $y^{\mathfrak{b}} : \underline{B}$, where $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$.

We need to form a purple term of $\underline{B} \otimes \underline{A}$, so 'split $\mathfrak{p}$ into $\mathfrak{b}$ and $\mathfrak{r}$'.

Then we can form $y \; {}_{\mathfrak{b}}{\otimes}_{\mathfrak{r}} \; x : \underline{B} \otimes \underline{A}$.

$$\mathsf{sym} :\equiv \lambda p.\mathsf{let}\ x \; {}_{\mathfrak{r}}{\otimes}_{\mathfrak{b}} \; y = p \ \mathsf{in}\ y \; {}_{\mathfrak{b}}{\otimes}_{\mathfrak{r}} \; x$$

□

But we don't have $\mathfrak{p} \prec \mathfrak{b} \otimes \mathfrak{r}$ literally, we need to allow for some symmetric monoidal structural rules.

Palette Splits

Need a more general judgement for when the palette linearly splits into two pieces: $\mathfrak{p} \prec \vec{\mathfrak{r}} \mathbin{\tilde{\otimes}} \vec{\mathfrak{b}}$

Symmetry: In palette $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$, we have a split

$$\mathfrak{p} \prec \mathfrak{b} \mathbin{\tilde{\otimes}} \mathfrak{r}$$

Associativity: In palette $\mathfrak{w} \prec (\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}) \otimes \mathfrak{y}$, we have a split

$$\mathfrak{w} \prec \mathfrak{r} \mathbin{\tilde{\otimes}} (\mathfrak{b} \otimes \mathfrak{y})$$

Cartesian weakening: In palette $\mathfrak{p} \prec (\mathfrak{r} \otimes \mathfrak{b}, \mathfrak{r}' \otimes \mathfrak{b}')$, we have a split

$$\mathfrak{p} \prec \mathfrak{r}' \mathbin{\tilde{\otimes}} \mathfrak{b}'$$

Rules for $\otimes$

Let $\mathfrak{p}$ be the top colour.

- Formation: If $\underline{A} : \mathcal{U}$ and $\underline{B} : \underline{A} \to \mathcal{U}$, then $\bigotimes_{(\underline{x}:\underline{A})} \underline{B}(\underline{x}) : \mathcal{U}$.
- Introduction: For any palette split $\mathfrak{p} \prec \vec{\mathfrak{r}} \,\tilde{\otimes}\, \vec{\mathfrak{b}}$ and terms $a : \underline{A}$ with colour $\mathfrak{r}$ and $b : \underline{B}(\underline{a})$ with colour $\mathfrak{b}$, there is a term

$$a \;{}_{\vec{\mathfrak{r}}}\!\otimes_{\vec{\mathfrak{b}}}\; b : \bigotimes_{(\underline{x}:\underline{A})} \underline{B}(\underline{x})$$

- Elimination: Any term $p : \bigotimes_{(\underline{x}:\underline{A})} \underline{B}(\underline{x})$ may be assumed to be of the form $x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y$ for some variables $x^{\mathfrak{r}} : \underline{A}, y^{\mathfrak{b}} : \underline{B}(\underline{x})$ with $\mathfrak{p} \prec \mathfrak{r} \otimes \mathfrak{b}$ in a term $c : C[x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y/z]$.

$$(\text{let } x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y = p \text{ in } c) : C[p/z]$$

- Computation: If the term really is of the form $a \;{}_{\vec{\mathfrak{r}'}}\!\otimes_{\vec{\mathfrak{b}'}}\; b$, then

$$(\text{let } x \;{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\; y = a \;{}_{\vec{\mathfrak{r}'}}\!\otimes_{\vec{\mathfrak{b}'}}\; b \text{ in } c) \equiv c[\vec{\mathfrak{r}'}/\mathfrak{r} \otimes \vec{\mathfrak{b}'}/\mathfrak{b} \mid a/x, b/x]$$

Eg. Associativity

Proposition

assoc : $\underline{A} \otimes (\underline{B} \otimes \underline{C}) \simeq (\underline{A} \otimes \underline{B}) \otimes \underline{C}$

Proof.

Use (derivable) triple inductions to define

$$\text{assoc} :\equiv \lambda p.\text{let } (a \,{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\, b) \,{}_{\mathfrak{p}}\!\otimes_{\mathfrak{y}}\, c = p \text{ in } a \,{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}\otimes\mathfrak{y}} (b \,{}_{\mathfrak{b}}\!\otimes_{\mathfrak{y}}\, c)$$

$$\text{associnv} :\equiv \lambda q.\text{let } a \,{}_{\mathfrak{r}}\!\otimes_{\mathfrak{g}} (b \,{}_{\mathfrak{b}}\!\otimes_{\mathfrak{y}}\, c) = q \text{ in } (a \,{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\, b) \,{}_{\mathfrak{r}\otimes\mathfrak{b}}\!\otimes_{\mathfrak{y}}\, c$$

Then to prove $\prod_{(p:(\underline{A}\otimes\underline{B})\otimes\underline{C})} \text{associnv}(\text{assoc}(p)) =_{(\underline{A}\otimes\underline{B})\otimes\underline{C}} p$, use induction again:

$$\text{linv} :\equiv \lambda p.\text{let } (a \,{}_{\mathfrak{r}}\!\otimes_{\mathfrak{b}}\, b) \,{}_{\mathfrak{p}}\!\otimes_{\mathfrak{y}}\, c = p \text{ in } \text{refl}_{(a\,{}_{\mathfrak{r}}\otimes_{\mathfrak{b}}\,b)\,{}_{\mathfrak{p}}\otimes_{\mathfrak{y}}\,c}$$

and similarly to show it is a right inverse. □

Eg. Associativity

Proposition

$\mathsf{assoc} : \underline{A} \otimes (\underline{B} \otimes \underline{C}) \simeq (\underline{A} \otimes \underline{B}) \otimes \underline{C}$

Proof.

Use (derivable) triple inductions to define

$$\mathsf{assoc} :\equiv \lambda p.\mathsf{let}\ (a \otimes b) \otimes c = p\ \mathsf{in}\ a \otimes (b \otimes c)$$
$$\mathsf{associnv} :\equiv \lambda q.\mathsf{let}\ a \otimes (b \otimes c) = q\ \mathsf{in}\ (a \otimes b) \otimes c$$

Then to prove $\prod_{(p:(\underline{A} \otimes \underline{B}) \otimes \underline{C})} \mathsf{associnv}(\mathsf{assoc}(p)) =_{(\underline{A} \otimes \underline{B}) \otimes \underline{C}} p$, use induction again:

$$\mathsf{linv} :\equiv \lambda p.\mathsf{let}\ (a \otimes b) \otimes c = p\ \mathsf{in}\ \mathsf{refl}_{(a \otimes b) \otimes c}$$

and similarly to show it is a right inverse. □

Eg. Associativity

Like Σ,

$$\mathsf{assoc} : \left(\sum_{(x:A)} \sum_{(y:B(x))} C(x)(y)\right) \simeq \left(\sum_{(v:\sum_{(x:A)} B(x))} C(\mathsf{pr}_1 v)(\mathsf{pr}_2 v)\right)$$

There is a dependent verison:

$$\mathsf{assoc} : \left(\bigodot_{(\underline{x}:\underline{A})} \bigodot_{(\underline{y}:\underline{B}(\underline{x}))} \underline{C}(\underline{x})(\underline{y})\right) \simeq \left(\bigodot_{(\underline{v}:\bigodot_{(\underline{x}:\underline{A})} \underline{B}(\underline{x}))} \mathsf{let}\ {\color{orange}x} \otimes {\color{blue}y} = \underline{v}\ \mathsf{in}\ \underline{C}(\underline{x})(\underline{y})\right)$$

Eg: Uniqueness principle for $\otimes$

Proposition

If $C : \bigotimes_{(\underline{x}:\underline{A})} \underline{B}(\underline{x}) \to \mathcal{U}$ *is a type family and* $f : \prod_{(p:\bigotimes_{(\underline{x}:\underline{A})} \underline{B}(\underline{x}))} C(p)$*, then for any* $p : A \otimes B$ *we have*

$$(\text{let } x \otimes y = p \text{ in } f(x \otimes y)) = f(p)$$

Proof.

By $\otimes$-induction we may assume $p \equiv x' \otimes y'$. Our goal is now

$$(\text{let } x \otimes y = x' \otimes y' \text{ in } f(x \otimes y)) = f(x' \otimes y')$$

Which by computation reduces to $f(x' \otimes y') = f(x' \otimes y')$, for which we have reflexivity. $\square$

(Cannot state this in indexed type or quantitative type theories)

Hom

$$\frac{\Gamma \times A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \otimes A \vdash B}{\Gamma \vdash A \multimap B}$$

Hom

$$\frac{\Gamma \times (x : A) \vdash B}{\Gamma \vdash \lambda x.b : \prod_{(x:A)} B}$$

$$\frac{\Gamma \otimes (y : \underline{A}) \vdash B}{\Gamma \vdash \partial y.b : \bigcirc\!\!\!\!\prod_{(y:\underline{A})} B}$$

Hom

$$\frac{\mathfrak{p} \mid \Gamma, x^{\mathfrak{p}} : A \vdash b : B}{\mathfrak{p} \mid \Gamma \vdash \lambda x.b : \prod_{(x:A)} B}$$

$$\frac{\mathfrak{w} \prec \mathfrak{p} \otimes \mathfrak{y} \mid \Gamma, y^{\mathfrak{y}} : \underline{A} \vdash b : B}{\mathfrak{p} \mid \Gamma \vdash \partial y.b : \bigcirc\!\!\!\!\prod_{(y^{\mathfrak{y}}:\underline{A})} B}$$

Rules for $\multimap$

Let $\mathfrak{p}$ be the top colour.

- Formation/Introduction: If $b : B$ is a term using a fresh assumption $x^{\mathfrak{y}} : \underline{A}$ in palette $\mathfrak{w} \prec \mathfrak{p} \otimes \mathfrak{y}$, for fresh colours $\mathfrak{w}$ and $\mathfrak{y}$, then there is a (purple) term

$$\partial^{\mathfrak{w}} x^{\mathfrak{y}}.b : \textstyle\coprod_{(x^{\mathfrak{y}}:\underline{A})} B$$

- Elimination: For any split $\mathfrak{p} \prec \vec{\mathfrak{r}} \,\tilde{\otimes}\, \vec{\mathfrak{b}}$ and terms $h : \coprod_{(x^{\mathfrak{y}}:\underline{A})} B$ with colour $\vec{\mathfrak{r}}$ and $a : \underline{A}$ with colour $\vec{\mathfrak{b}}$, there is a term

$$h_{\vec{\mathfrak{r}}}\langle a\rangle_{\vec{\mathfrak{b}}} : B[\![\vec{\mathfrak{b}}/\mathfrak{y} \mid a/x]\!]$$

- Computation: $(\partial^{\circ} x^{\mathfrak{y}}.b)_{\vec{\mathfrak{r}}}\langle a\rangle_{\vec{\mathfrak{b}}} \equiv b[\![\vec{\mathfrak{b}}/\mathfrak{y} \mid a/x]\!]$
- Uniqueness: $h \equiv \partial^{\mathfrak{w}} x^{\mathfrak{y}}.(h_{\mathfrak{p}}\langle x\rangle_{\mathfrak{y}})$

Eg. Currying

Let $\mathfrak{y}$ be the top colour.

Proposition

There is a map $((\underline{A} \otimes \underline{B}) \multimap \underline{C}) \to (\underline{A} \multimap (\underline{B} \multimap \underline{C}))$.

Proof.

Suppose $h^{\mathfrak{y}} : (A \otimes B) \multimap C$. Using ∂-abstraction binds $x^{\mathfrak{r}} : \underline{A}$, and our goal is $\underline{B} \multimap \underline{C}$ in $\mathfrak{o} \prec \mathfrak{y} \otimes \mathfrak{r}$.

Another ∂-abstraction binds $y^{\mathfrak{b}} : \underline{B}$, and our goal is $\underline{C}$ in $\mathfrak{w} \prec (\mathfrak{o} \prec \mathfrak{y} \otimes \mathfrak{r}) \otimes \mathfrak{b}$.

Pairing x and y gives a term $x\ {}_{\mathfrak{r}}{\otimes}_{\mathfrak{b}}\ y : \underline{A} \otimes \underline{B}$ of colour $\mathfrak{r} \otimes \mathfrak{b}$.
Applying h to this gives $h_{\mathfrak{y}}\langle x\ {}_{\mathfrak{r}}{\otimes}_{\mathfrak{b}}\ y\rangle_{\mathfrak{r}\otimes\mathfrak{b}} : \underline{C}$.

$\lambda h.\partial^{\mathfrak{o}} x^{\mathfrak{r}}.\partial^{\mathfrak{w}} y^{\mathfrak{b}}.h_{\mathfrak{y}}\langle x\ {}_{\mathfrak{r}}{\otimes}_{\mathfrak{b}}\ y\rangle_{\mathfrak{r}\otimes\mathfrak{b}}$ or simply $\lambda h.\partial x.\partial y.h\langle x \otimes y\rangle$

□

Eg. Homs vs Functions

When can we build a non-trivial map? (Here meaning a map that does not use any variable marked.)

Given $f : \underline{A} \to \underline{B}$	$\underline{A} \to \underline{B}$	$\underline{A} \multimap \underline{B}$	$\underline{A} \to \underline{B} \times \underline{B}$	$\underline{A} \multimap \underline{B} \times \underline{B}$	$\underline{A} \to \underline{B} \otimes \underline{B}$	$\underline{A} \multimap \underline{B} \otimes \underline{B}$
Given $h : \underline{A} \multimap \underline{B}$	$\underline{A} \to \underline{B}$	$\underline{A} \multimap \underline{B}$	$\underline{A} \to \underline{B} \times \underline{B}$	$\underline{A} \multimap \underline{B} \times \underline{B}$	$\underline{A} \to \underline{B} \otimes \underline{B}$	$\underline{A} \multimap \underline{B} \otimes \underline{B}$
Given $o : 1$	$\underline{A} \to \underline{A}$	$\underline{A} \multimap \underline{A}$	$\underline{A} \to 1$	$\underline{A} \multimap 1$	$\underline{A} \to \mathbb{S}$	$\underline{A} \multimap \mathbb{S}$
Given $s : \mathbb{S}$	$\underline{A} \to \underline{A}$	$\underline{A} \multimap \underline{A}$	$\underline{A} \to 1$	$\underline{A} \multimap 1$	$\underline{A} \to \mathbb{S}$	$\underline{A} \multimap \mathbb{S}$

Hom Extensionality

Hom Extensionality

Let us write the top colour as $\mathfrak{r}$. For $f, g : \textcircled{\textstyle\prod}_{(x:\underline{A})} B\langle x\rangle$,

$$\mathsf{homapp}(f, g) : (f = g) \to \textcircled{\textstyle\prod}_{(x:\underline{A})} f\langle x\rangle = g\langle x\rangle$$

is given by path induction:

$$\mathsf{homapp}(f, f)(\mathsf{refl}_f) :\equiv \partial x.\mathsf{refl}_{f\langle x\rangle}$$

Axiom Homext

For any $f, g : \textcircled{\textstyle\prod}_{(x:\underline{A})} B\langle x\rangle$, the function $\mathsf{homapp}(f, g)$ is an equivalence.

Theorem

Univalence implies hom extensionality.

Strategy

Almost the same proof as for functions! Following the HoTT book:

1. 'Naive' homext (there is a map back).
2. Weak homext (homs into contractible families are contractible).
3. Homext.

Quick Lemma

Definition

The postcomposition of $h : A \to B$ with $f : B \to B'$ is defined by

$$\mathsf{postcomp}(f, h) : A \to B'$$
$$\mathsf{postcomp}(f, h) :\equiv \lambda x.f(h(x))$$

Lemma

Any equivalence $e : B \simeq B'$ induces an equivalence $(A \to B) \simeq (A \to B')$ by postcomposition with e.

Proof.

e is the image of some $p : B = B'$ under univalence. By path induction, assume $p \equiv \mathsf{refl}_B$, so $e \equiv \mathsf{id}_B$. Then postcomposition with e is the identity, and so is an equivalence. □

Quick Lemma

Definition

The postcomposition of $h : \underline{A} \multimap \underline{B}$ with $\underline{f} : \underline{B} \to \underline{B}'$ is defined by

$$\begin{aligned} &\mathsf{postcomp}(\underline{f}, h) : \underline{A} \multimap \underline{B}' \\ &\mathsf{postcomp}(\underline{f}, h) :\equiv \partial x. \underline{f}(h\langle x \rangle) \end{aligned}$$

Lemma

Any equivalence $e : B \simeq B'$ induces an equivalence $(\underline{A} \multimap \underline{B}) \simeq (\underline{A} \multimap \underline{B}')$ by postcomposition with $\underline{e}$.

Proof.

e is the image of some $p : B = B'$ under univalence. By path induction, assume $p \equiv \mathsf{refl}_B$, so $e \equiv \mathsf{id}_B$. Then postcomposition with $\underline{e}$ is the identity, and so is an equivalence. □

Naive Funext

Proposition

For $f, g : A \to B$ there is a map $\left(\prod_{(x:A)} f(x) = g(x)\right) \to (f = g)$.

Proof.

Given $h : \prod_{(x:A)} f(x) = g(x)$, define

$$
\begin{aligned}
d, e &: A \to \left(\sum_{(y:B)} \sum_{(y':B)} y = y'\right) \\
d &:\equiv \lambda x.(f(x), f(x), \mathsf{refl}_{f(x)}) \\
e &:\equiv \lambda x.(f(x), g(x), h(x))
\end{aligned}
$$

Then $d = e$ because they become equal under the equivalence

$$\mathsf{postcomp}(\mathsf{pr}_1, -) : \left[A \to \left(\sum_{(y:B)} \sum_{(y':B)} y = y'\right)\right] \to [A \to B]$$

And ap of $\mathsf{postcomp}(\mathsf{pr}_2, -)$ on the path $d = e$ gives a path $\lambda x.f(x) = \lambda x.g(x)$, which is $f = g$. □

Naive Homext

Proposition

For $f, g : \underline{A} \multimap \underline{B}$ there is a map $\left(ⓟ_{(x:\underline{A})} f\langle x\rangle = g\langle x\rangle\right) \to (f = g)$.

Proof.

Given $h : ⓟ_{(x:\underline{A})} f\langle x\rangle = g\langle x\rangle$, define

$$d, e : \underline{A} \multimap \left(\sum_{(y:\underline{B})}\sum_{(y':\underline{B})} y = y'\right)$$
$$d :\equiv \partial x.(f\langle x\rangle, f\langle x\rangle, \mathsf{refl}_{f\langle x\rangle})$$
$$e :\equiv \partial x.(f\langle x\rangle, g\langle x\rangle, h\langle x\rangle)$$

Then $d = e$ because they become equal under the equivalence

$$\mathsf{postcomp}(\mathsf{pr}_1, -) : \left[\underline{A} \multimap \left(\sum_{(y:\underline{B})}\sum_{(y':\underline{B})} y = y'\right)\right] \to [\underline{A} \multimap \underline{B}]$$

And ap of $\mathsf{postcomp}(\mathsf{pr}_2, -)$ on the path $d = e$ gives a path, $\partial x.f\langle x\rangle = \partial x.g\langle x\rangle$, which is $f = g$. □

Weak Funext

Proposition

$\prod_{(x:A)} \text{isContr}(B(x)) \to \text{isContr}\left(\prod_{(x:A)} B(x)\right)$

Proof.

Suppose $w : \prod_{(x:A)} \text{isContr}(B(x))$. From w and univalence we can build a term of $\prod_{(x:A)}(B(x) = 1)$.

Then naive funext gives $p : B = (\lambda x.1)$, and we can form

$$\text{ap}_{\prod_{(x:A)} -(x)}(p) : \left(\prod_{(x:A)} B(x)\right) = (A \to 1)$$

Now $A \to 1$ is contractible because for any $f : A \to 1$ we have $f \equiv \lambda x.f(x) \equiv \lambda x.\star$. Transport $\text{isContr}(A \to 1)$ along the above path. □

Weak Homext

Proposition

$\textcircled{\prod}_{(x:\underline{A})} \text{isContr}(B\langle x\rangle) \to \text{isContr}\left(\textcircled{\prod}_{(x:\underline{A})} B\langle x\rangle\right)$

Proof.

Suppose $w : \textcircled{\prod}_{(x:\underline{A})} \text{isContr}(B\langle x\rangle)$. From w and univalence we can build a term of $\textcircled{\prod}_{(x:\underline{A})}(B\langle x\rangle = 1)$.
Then naive homext gives $p : B = (\partial x.1)$, and we can form

$$\text{ap}_{\textcircled{\prod}_{(x:\underline{A})} -\langle x\rangle}(p) : \left(\textcircled{\prod}_{(x:\underline{A})} B\langle x\rangle\right) = (\underline{A} \multimap 1)$$

Now $\underline{A} \multimap 1$ is contractible because for any $f : \underline{A} \multimap 1$ we have $f \equiv \partial x.f\langle x\rangle \equiv \partial x.\star$. Transport $\text{isContr}(\underline{A} \multimap 1)$ along the above path. □

$\to$ Preserves Σ

Proposition

$$A \to (B \times C) \simeq (A \to B) \times (A \to C)$$

Or with maximal dependency:

$$\prod_{(x:A)} \sum_{(y:B(x))} C(x)(y) \simeq \sum_{(g:\prod_{(x:A)} B(x))} \prod_{(x:A)} C(x)(g(x))$$

Proof.

Define maps back and forth:

$$\begin{aligned} f &\mapsto (\lambda x.\mathsf{pr}_1(f(x)), \lambda x.\mathsf{pr}_2(f(x))) \\ (g, h) &\mapsto \lambda x.(g(x), h(x)) \end{aligned}$$

Both round-trips are definitionally the identity. □

$\multimap$ Preserves Σ

Proposition

$$\underline{A} \multimap (\underline{B} \times \underline{C}) \simeq (\underline{A} \multimap \underline{B}) \times (\underline{A} \multimap \underline{C})$$

Or with maximal dependency:

$$⨿_{(x:\underline{A})} \sum_{(y:B\langle x \rangle)} C\langle x \rangle(y) \simeq \sum_{(g:⨿_{(x:\underline{A})} B\langle x \rangle)} ⨿_{(x:\underline{A})} C\langle x \rangle(g\langle x \rangle)$$

Proof.

Define maps back and forth:

$$f \mapsto (\partial x.\mathsf{pr}_1(f\langle x \rangle), \partial x.\mathsf{pr}_2(f\langle x \rangle))$$
$$(g, h) \mapsto \partial x.(g\langle x \rangle, h\langle x \rangle)$$

Both round-trips are definitionally the identity. □

Theorem

Function extensionality holds.

Proof.

Fixing an f and working fibrewise, we need

$$\left(\sum_{(g:\prod_{(x:A)} B(x))}(f = g)\right) \to \left(\sum_{(g:\prod_{(x:A)} B(x))}\prod_{(x:A)} f(x) = g(x)\right)$$

given by $\lambda(g,p).(g, \mathsf{happly}(f,g)(p))$ is an equivalence. The LHS is contractible, so we just need the RHS also contractible. By the last Proposition, the RHS is equivalent to

$$\prod_{(x:A)}\sum_{(y:B(x))} f(x) = y$$

which is contractible by weak homext. □

Funext

Theorem

Hom extensionality holds.

Proof.

Fixing an f and working fibrewise, we need

$$\left(\sum_{(g:\textstyle\prod_{(x:\underline{A})} B\langle x\rangle)}(f = g)\right) \to \left(\sum_{(g:\textstyle\prod_{(x:\underline{A})} B\langle x\rangle)} \textstyle\prod_{(x:\underline{A})} f\langle x\rangle = g\langle x\rangle\right)$$

given by $\lambda(g,p).(g, \mathrm{homapp}(f,g)(p))$ is an equivalence. The LHS is contractible, so we just need the RHS also contractible. By the last Proposition, the RHS is equivalent to

$$\textstyle\prod_{(x:\underline{A})}\sum_{(y:B\langle x\rangle)} f\langle x\rangle = y$$

which is contractible by weak funext. □

Summary Future

- Extension of HoTT with $\natural$, $\otimes$, $\multimap$ and $\mathbb{S}$
- Compatible with existing synthetic results
- Can show: a map of spaces $\underline{X} \to \underline{Y}$ gives a 'six functor formalism' between $\underline{X} \to \mathrm{Spec}$ and $\underline{Y} \to \mathrm{Spec}$
- What can we prove about (co)homology synthetically?
- Can generalise to let $\mathcal{C}$ be not pointed? (Probably!)

References I

Atkey, Robert (2018). "Syntax and semantics of quantitative type theory". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '18. Oxford, United Kingdom: Association for Computing Machinery, pp. 56–65. ISBN: 978-1-4503-5583-4. DOI: 10.1145/3209108.3209189.

Fu, Peng, Kohei Kishida, and Peter Selinger (2020). "Linear Dependent Type Theory for Quantum Programming Languages: Extended Abstract". In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '20. Saarbrücken, Germany: Association for Computing Machinery, pp. 440–453. ISBN: 978-1-4503-7104-9. DOI: 10.1145/3373718.3394765.

Hoyois, Marc (2019). "Topoi of parametrized objects". In: *Theory and Applications of Categories* 34, Paper No. 9, 243–248. URL: http://www.tac.mta.ca/tac/volumes/34/9/34-09abs.html.

Isaev, Valery (2021). "Indexed type theories". In: *Mathematical Structures in Computer Science* 31.1, pp. 3–63. DOI: 10.1017/S0960129520000092.

References II

Krishnaswami, Neelakantan R., 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*. POPL '15. Mumbai, India: Association for Computing Machinery, pp. 17–30. ISBN: 978-1-4503-3300-9. DOI: 10.1145/2676726.2676969.

McBride, Conor (2016). "I got plenty o' nuttin'". In: *A list of successes that can change the world*. Vol. 9600. Lecture Notes in Computer Science. Springer International Publishing, pp. 207–233. DOI: 10.1007/978-3-319-30936-1_12.

Moon, Benjamin, Harley Eades III, and Dominic Orchard (2021). "Graded Modal Dependent Type Theory". In: *Programming Languages and Systems*. Ed. by Nobuko Yoshida. Cham: Springer International Publishing, pp. 462–490. DOI: 10.1007/978-3-030-72019-3_17.

O'Hearn, Peter (2003). "On bunched typing". In: *Journal of Functional Programming* 13.4, pp. 747–796. ISSN: 0956-7968. DOI: 10.1017/S0956796802004495.

References III

O'Hearn, Peter W. and David J. Pym (1999). "The logic of bunched implications". In: *Bulletin of Symbolic Logic* 5.2, pp. 215–244. ISSN: 1079-8986. DOI: 10.2307/421090.

Riley, Mitchell, Eric Finster, and Daniel R. Licata (2021). *Synthetic Spectra via a Monadic and Comonadic Modality*. arXiv: 2102.04099.

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