# Extending Homotopy Type Theory with Linear Type Formers

Mitchell Riley[1]
jww. Dan Licata[1]
Eric Finster[2]

Wesleyan University[1]
University of Cambridge[2]

26th March 2021

# Homotopy Type Theory

# Our Setting

Martin-Löf Type Theory with:

- Dependent pair types $(x : A) \times B$
- Dependent function types $(x : A) \to B$
- Empty type $0$
- Unit type $1$
- Equality types $a =_A a'$
- Universe $\mathcal{U}$

# Equality Types

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

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

- Intro: There is always a reflexivity equality.
- Elim: If type depending on a generic equality is inhabited for reflexivity, it's inhabited for any equality.

# Equality of Pairs

Equality of pairs can be described explicitly.

## Lemma

*$(a, b) =_{(x:A)\times B} (a', b')$ is equivalent to*

$$(p : a =_A a') \times (b =_{B(a)} p^* b')$$

*where $p^* : B(a') \to B(a)$ is defined using $=$-elim.*

# Equality of Pairs

Equality of pairs can be described explicitly.

## Lemma

*$(a, b) =_{(x:A)\times B} (a', b')$ is equivalent to*

$$(p : a =_A a') \times (b =_{B(a)} p^* b')$$

*where $p^* : B(a') \to B(a)$ is defined using $=$-elim.*

## Definition

A function $f : A \to B$ is an equivalence if it has a left inverse and right inverse (up to $=$). Write $A \simeq B$ for a pair of a map $A \to B$ and a proof it is an equivalence.

# Equality of Functions

We can't prove anything interesting about equality of functions.

# Equality of Functions

We can't prove anything interesting about equality of functions.
Is there a candidate type for $f =_{(x:A)\to B} g$ to be equivalent to?

# Equality of Functions

We can't prove anything interesting about equality of functions.
Is there a candidate type for $f =_{(x:A)\to B} g$ to be equivalent to?
Yes!

$$(\lambda p.\mathsf{let}\ \mathsf{refl}_h = p\ \mathsf{in}\ \lambda x.\mathsf{refl}_{h(x)}) \\ : (f = g) \to \big((x : A) \to f(x) =_{B(x)} g(x)\big)$$

# Equality of Functions

We can't prove anything interesting about equality of functions.
Is there a candidate type for $f =_{(x:A)\to B} g$ to be equivalent to?
Yes!

$$(\lambda p.\mathsf{let}\ \mathsf{refl}_h = p\ \mathsf{in}\ \lambda x.\mathsf{refl}_{h(x)}) \\ : (f = g) \to \big((x : A) \to f(x) =_{B(x)} g(x)\big)$$

## Axiom (Function Extensionality)

*This map is an equivalence.*

# Equality of Functions

We can't prove anything interesting about equality of functions.
Is there a candidate type for $f =_{(x:A)\to B} g$ to be equivalent to?
Yes!

$$(\lambda p.\mathsf{let}\ \mathsf{refl}_h = p\ \mathsf{in}\ \lambda x.\mathsf{refl}_{h(x)}) \\ : (f = g) \to \big((x : A) \to f(x) =_{B(x)} g(x)\big)$$

## Axiom (Function Extensionality)

*This map is an equivalence.*

Asserting this axiom restricts the interpretations of the type theory. (But sets still work!)

# Equality of Types

What about the universe?

# Equality of Types

What about the universe? Another guess:

$$(\lambda p.\mathsf{let}\ \mathsf{refl}_C = p\ \mathsf{in}\ \mathsf{id}_C) : (A = B) \to (A \simeq B)$$

# Equality of Types

What about the universe? Another guess:

$$(\lambda p.\mathsf{let}\ \mathsf{refl}_C = p\ \mathsf{in}\ \mathsf{id}_C) : (A = B) \to (A \simeq B)$$

## Axiom (Univalence)

*This map is an equivalence.*

# Uniqueness of Identity Proofs?

In sets, the type $a =_A a'$ is given by $\{\star\}$ if the element $a$ is the same element as $a'$, and $\varnothing$ otherwise.

# Uniqueness of Identity Proofs?

In sets, the type $a =_A a'$ is given by $\{\star\}$ if the element $a$ is the same element as $a'$, and $\varnothing$ otherwise.

## Axiom (Uniqueness of Identity Proofs)

*For any $p, p' : a =_A a'$, there is a term of $p = p'$.*

# Uniqueness of Identity Proofs?

In sets, the type $a =_A a'$ is given by $\{\star\}$ if the element $a$ is the same element as $a'$, and $\varnothing$ otherwise.

## Axiom (Uniqueness of Identity Proofs)

*For any $p, p' : a =_A a'$, there is a term of $p = p'$.*

UIP and Univalence are incompatible:

## Proof.

There are two equivalences $\mathsf{id}, \mathsf{swap} : \mathsf{Bool} \simeq \mathsf{Bool}$. Univalence turns these into equalities $\mathsf{ua}(\mathsf{id}), \mathsf{ua}(\mathsf{id}) : \mathsf{Bool} = \mathsf{Bool}$, and UIP claims these equalities are equal. So $\mathsf{ua}(\mathsf{id}) = \mathsf{ua}(\mathsf{swap})$ which means $\mathsf{id} = \mathsf{swap}$, but then

$$\mathsf{true} = \mathsf{id}(\mathsf{true}) = \mathsf{swap}(\mathsf{true}) = \mathsf{false},$$

uh oh. □

# Proof Relevance

$$f : a =_A b$$
$$g : a =_A b$$
$$k : a =_A b$$

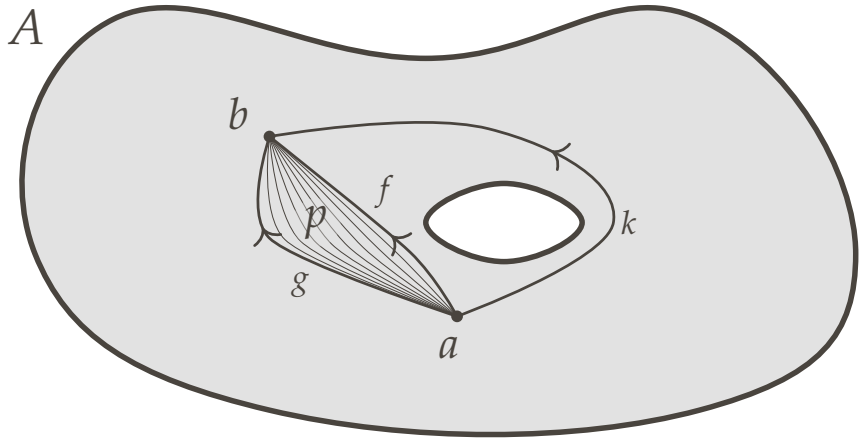



Diagram stolen from Pelayo and Warren

# Proof Relevance

$$f : a =_A b$$
$$g : a =_A b$$
$$k : a =_A b$$
$$p : f =_{a =_A b} g$$

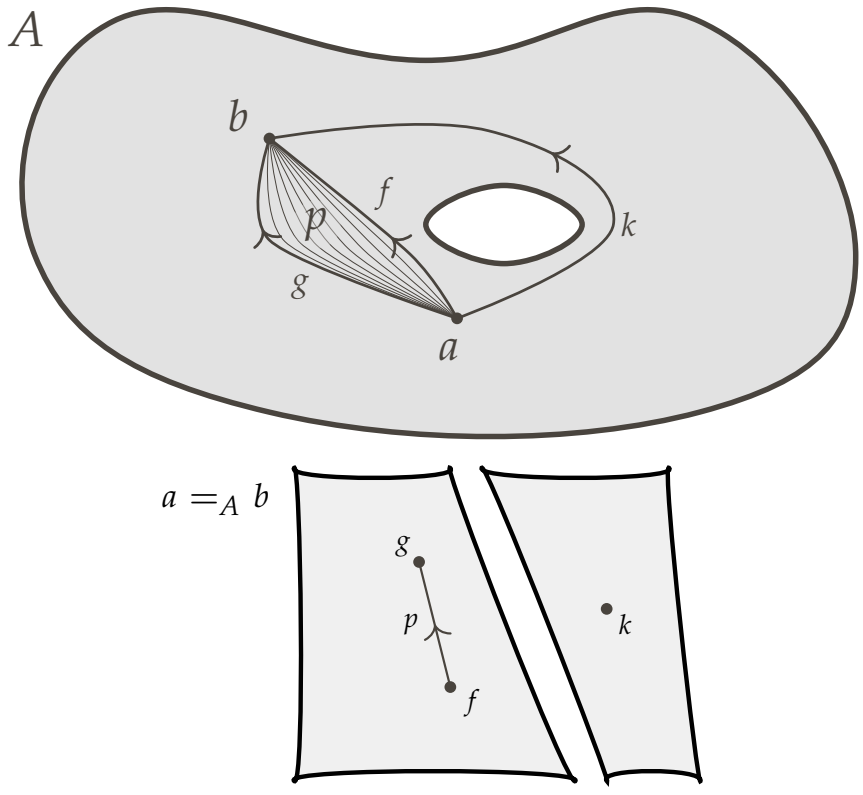



Diagram stolen from Pelayo and Warren

# Homotopical Interpretations

**Theorem (Kapulkin and Lumsdaine 2012, after Voevodsky)**

*HoTT, meaning (MLTT + Univalence + HITs), has an interpretation in 'simplicial sets'.*

which are a model of the homotopy theory of spaces. I think of these as 'sets with possibly complicated equalities'.

# Homotopical Interpretations

**Theorem (Kapulkin and Lumsdaine 2012, after Voevodsky)**

*HoTT, meaning (MLTT + Univalence + HITs), has an interpretation in 'simplicial sets'.*

which are a model of the homotopy theory of spaces. I think of these as 'sets with possibly complicated equalities'.

Homotopy (Type Theory) $\leftrightarrow$ (Homotopy Type) Theory

# Working Synthetically

| **Result** | **Date** | **Synthetic Version** |
|---|---|---|
| $\pi_1(S^1) = \mathbb{Z}$ | 1892 | Licata, Shulman |
| Freudenthal Suspension Theorem | 1937 | Lumsdaine |
| Blakers–Massey Theorem | 1949 | Favonia, Finster, Licata, Lumsdaine |
| James Construction | 1955 | Brunerie |
| Localisation of Spaces | 1970 | Christensen, Opie, Rijke, Scoccola |

Can be formalised *and* apply in all models.

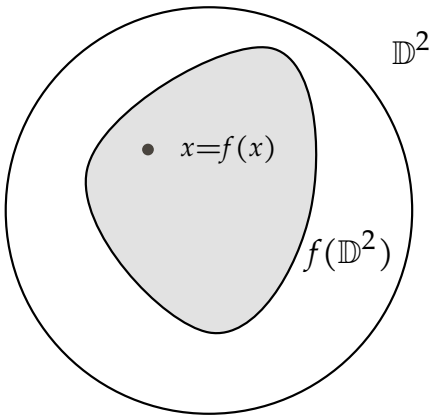
# Modal Homotopy Type Theory

# Brouwer's Fixed Point Theorem

Sometimes working up to homotopy is not enough.

## Theorem (Brouwer)

*Let $\mathbb{D}^2$ denote the unit disc. Any continuous map $f : \mathbb{D}^2 \to \mathbb{D}^2$ has a fixed point.*

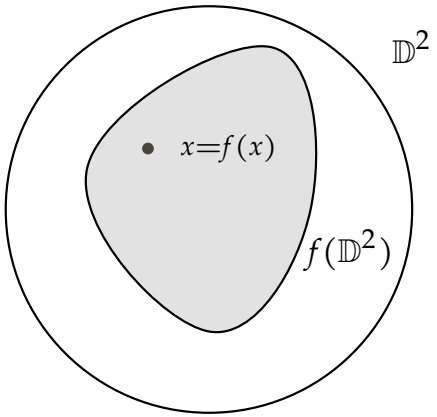


# Brouwer's Fixed Point Theorem

Sometimes working up to homotopy is not enough.

## Theorem (Brouwer)

*Let $\mathbb{D}^2$ denote the unit disc. Any continuous map $f : \mathbb{D}^2 \to \mathbb{D}^2$ has a fixed point.*




Up to homotopy, $\mathbb{D}^2$ is equivalent to 1. But then the theorem is contentless: of course any map $1 \to 1$ has a fixed point.

We need a way of thinking about $\mathbb{D}^2$ as a topological space made up of distinct points that are not equal to each other.

# Change the Interpretation

Rather than interpreting types as homotopy types
(= sets with complicated equality),
we can interpret them as 'topological $\infty$-groupoids'
(= sets with complicated equality AND a topology)

# Change the Interpretation

Rather than interpreting types as homotopy types
(= sets with complicated equality),
we can interpret them as 'topological $\infty$-groupoids'
(= sets with complicated equality AND a topology)

| | Trivial Equality | Interesting Equality |
|---|---|---|
| Trivial Topology | $1, \mathbb{N}$ | |
| Interesting Topology | $\mathbb{D}^2 \subset \mathbb{R}^2, \mathbb{S}^1 \subset \mathbb{R}^2$ | $\mathcal{U}$ |

# Accessing the New Structure

Cohesive Type Theory (Shulman 2018) adds new unary type formers:

- $\sharp A$ retopologises $A$ with the codiscrete topology
- $\flat A$ retopologises $A$ with the discrete topology
- $\int A$ turns *topological paths* in $A$ into *equalities*

# Accessing the New Structure

Cohesive Type Theory (Shulman 2018) adds new unary type formers:

- $\sharp A$ retopologises $A$ with the codiscrete topology
- $\flat A$ retopologises $A$ with the discrete topology
- $\int A$ turns *topological paths* in $A$ into *equalities*

| | Trivial Equality | Interesting Equality |
|---|---|---|
| Trivial Topology | $1, \mathbb{N}, \int \mathbb{D}^2$ | $\int \mathbb{S}^1$ |
| Interesting Topology | $\mathbb{D}^2 \subset \mathbb{R}^2, \mathbb{S}^1 \subset \mathbb{R}^2$ | $\mathcal{U}$ |

# Rules for $\flat$

Extremely roughly:

$$\flat\text{-INTRO}\ \frac{\Gamma \vdash a : A}{\mathsf{f}(\Gamma) \vdash a^{\flat} : \flat A}$$

$$\flat\text{-ELIM}\ \frac{\begin{array}{c}\Gamma, z : \flat A \vdash C\ \mathsf{type} \\ \Gamma \vdash s : \flat A \qquad \Gamma, \mathsf{f}(x : A) \vdash c : C[x^{\flat}/z]\end{array}}{\Gamma \vdash \mathsf{let}\ x^{\flat} = s\ \mathsf{in}\ c : C[s/z]}$$

Much like $\Box$ in the style of (Pfenning and Davies 2001)
Or ! in Dual Intuitionistic Linear Logic by (Barber 1996)

# Plan

The plan is:

1. Think of a model of HoTT with more structure than bare homotopy types,
2. Add some semantic operations as context formers,
3. Add new type formers internalising the context formers.

Our goal is to *extend* HoTT so we have access to all the synthetic results proven so far.

# Homotopy Type Theory and Linearity

# Stable Homotopy Theory

An important tool in homotopy theory is 'cohomology'. For any space $X$, there is a sequence of groups $H^n(X)$ containing information about the space.

There are many kinds of cohomology $H$, and each kind can be packaged into a 'spectrum'. A spectrum is a proof relevant version of an abelian group.

# Stable Homotopy Theory

An important tool in homotopy theory is 'cohomology'. For any space $X$, there is a sequence of groups $H^n(X)$ containing information about the space.

There are many kinds of cohomology $H$, and each kind can be packaged into a 'spectrum'. A spectrum is a proof relevant version of an abelian group.

## Definition

A *spectrum* $E$ is a sequence of pointed types $E : \mathbb{N} \to \mathcal{U}_\star$ together with pointed equivalences $\alpha_n : E_n \to_\star \Omega E_{n+1}$, where $\Omega(A, \star) :\equiv (\star =_A \star)$.

$E_0 \simeq \Omega E_1$, so $E_0$ is a group. $E_0 \simeq \Omega\Omega E_2$, so $E_0$ is abelian. $E_0 \simeq \Omega\Omega\Omega E_3$, so the commutativity of $E_0$ satisfies one level of coherence ...

# Spectra in Type Theory

We could use the above definition of spectra and manipulate them using type theory. But that's hard!

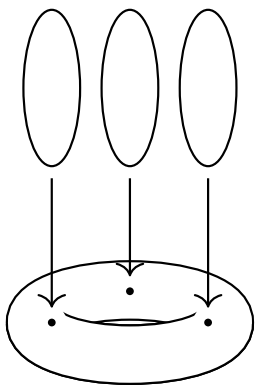
The category of spectra is a model of Intuitionistic Linear Logic with $(\otimes, \multimap, \& = \oplus)$. But that is not very expressive compared with HoTT!

# Parameterised Spectra

## Definition

A *parameterised spectrum* is a $\infty$-groupoid indexed family of spectra.

Translation: a set-with-equality, so that every element has an associated abelian-group-with-equality.

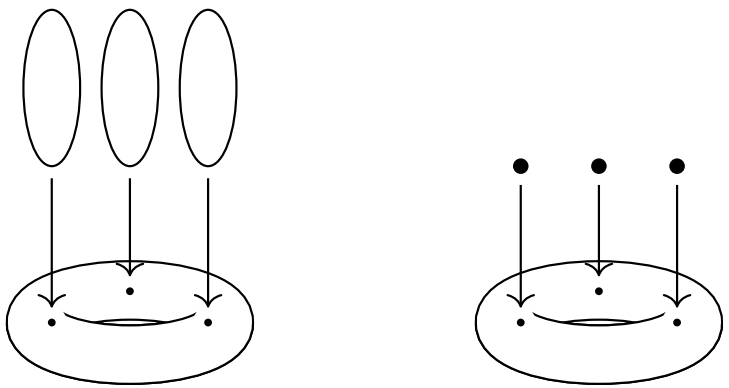


# Parameterised Spectra

## Definition

A *parameterised spectrum* is a $\infty$-groupoid indexed family of spectra.

Translation: a set-with-equality, so that every element has an associated abelian-group-with-equality.

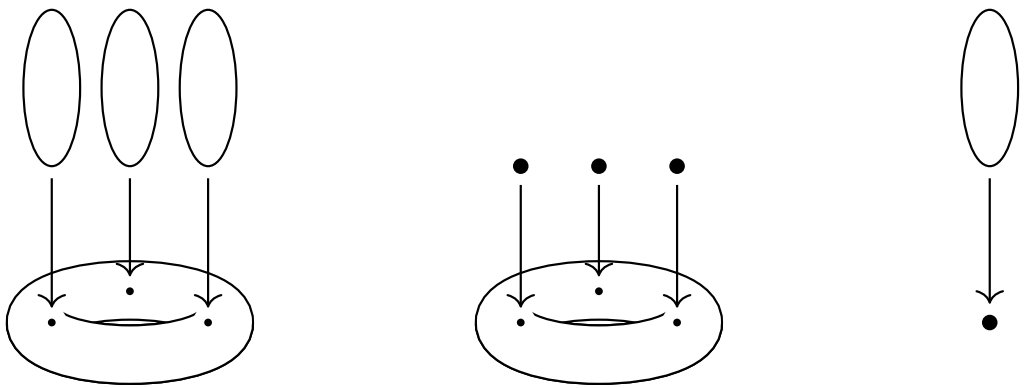


# Parameterised Spectra

## Definition

A *parameterised spectrum* is a $\infty$-groupoid indexed family of spectra.

Translation: a set-with-equality, so that every element has an associated abelian-group-with-equality.

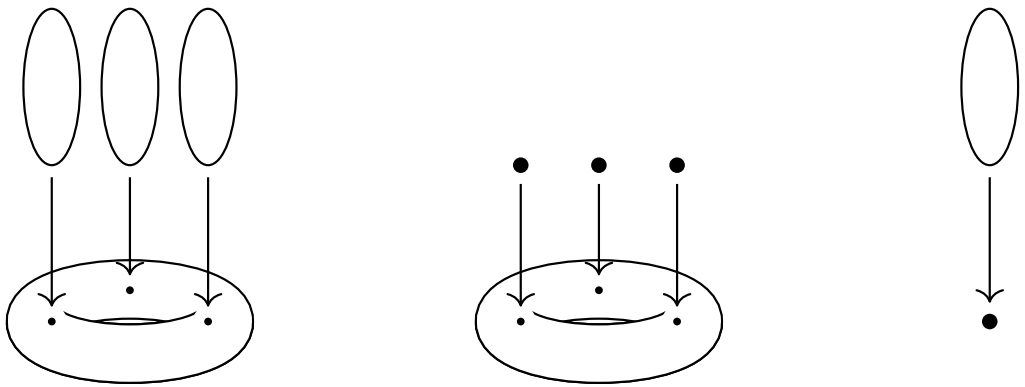


# Parameterised Spectra

## Definition

A *parameterised spectrum* is a $\infty$-groupoid indexed family of spectra.

Translation: a set-with-equality, so that every element has an associated abelian-group-with-equality.

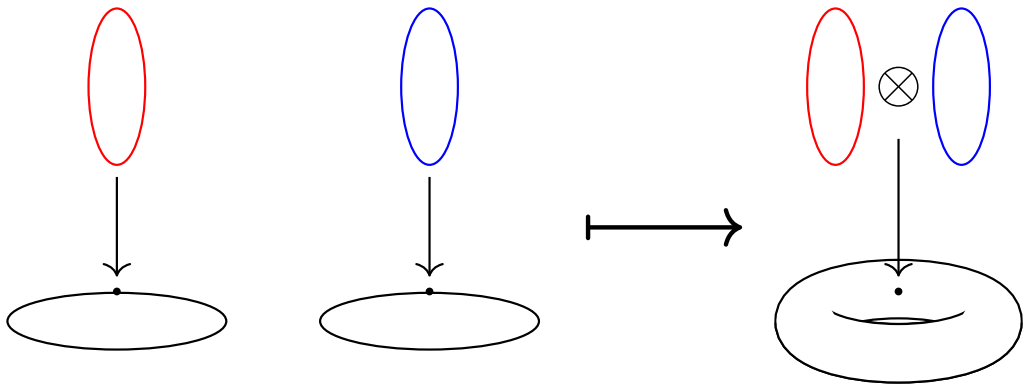


## Theorem (Joyal 2008 plus Shulman 2019)

*MLTT + Univalence can be interpreted in parameterised spectra.*

# Tensor Product

For two types $A$ and $B$ there should be a type $A \otimes B$ that calculates the 'external tensor product'.



This is a symmetric monoidal product with no additional structural rules, like the $\otimes$ of linear logic.

# Bunched Contexts

We can take a cue from 'bunched logics' (O'Hearn and Pym 1999), where there are two ways of combining contexts, an ordinary cartesian one and a linear one.

$$\frac{\Gamma_1 \text{ ctx} \qquad \Gamma_2 \text{ ctx}}{\Gamma_1, \Gamma_2 \text{ ctx}} \qquad\qquad \frac{\Gamma_1 \text{ ctx} \qquad \Gamma_2 \text{ ctx}}{(\Gamma_1)(\Gamma_2) \text{ ctx}}$$

For the comma *only*, we have weakening and contraction as normal.

(We are ignoring dependency in $\otimes$ for now)

# Bunched Contexts

A typical context:

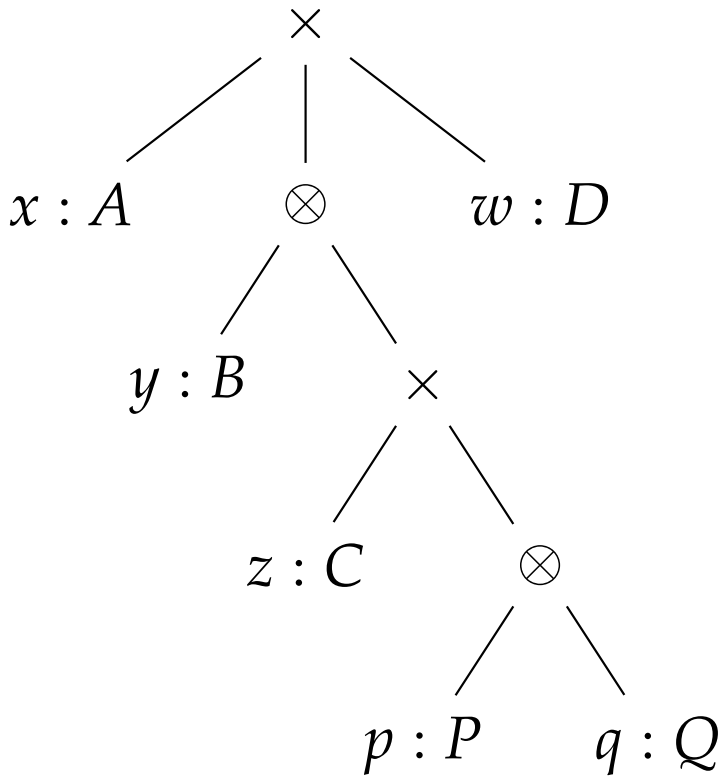
$$x : A, (y : B)(z : C, (p : P)(q : Q)), w : D$$

# Bunched Contexts

A typical context:

$$x : A, (y : B)(z : C, (p : P)(q : Q)), w : D$$

Or as a tree:



# Instead: Palettes

This gets very confusing very fast. Instead, we track the tree structure of the context separately, in a 'palette'.

$$(x : A)(y : B), z : C, w : D$$

becomes

$$\mathfrak{r} \otimes \mathfrak{b} \mid x^{\mathfrak{r}} : A, y^{\mathfrak{b}} : B, z^{\top} : C, w^{\top} : D$$

# Instead: Palettes

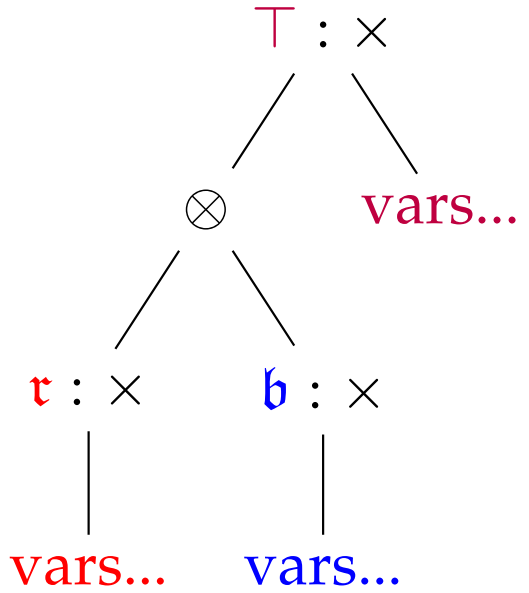
This gets very confusing very fast. Instead, we track the tree structure of the context separately, in a 'palette'.

$$(x : A)(y : B), z : C, w : D$$

becomes

$$\mathfrak{r} \otimes \mathfrak{b} \mid x^{\mathfrak{r}} : A, y^{\mathfrak{b}} : B, z^{\top} : C, w^{\top} : D$$

where the palette $\mathfrak{r} \otimes \mathfrak{b}$ on its own represents



# Instead: Palettes

From earlier,

$$x : A, (y : B)(z : C, (p : P)(q : Q)), w : D$$

becomes

$$\mathfrak{r} \otimes (\mathfrak{g} \prec \mathfrak{b} \otimes \mathfrak{y}) \mid x^{\top} : A, y^{\mathfrak{r}} : B, z^{\mathfrak{g}} : C, p^{\mathfrak{b}} : P, p^{\mathfrak{y}} : Q, w^{\top} : D$$

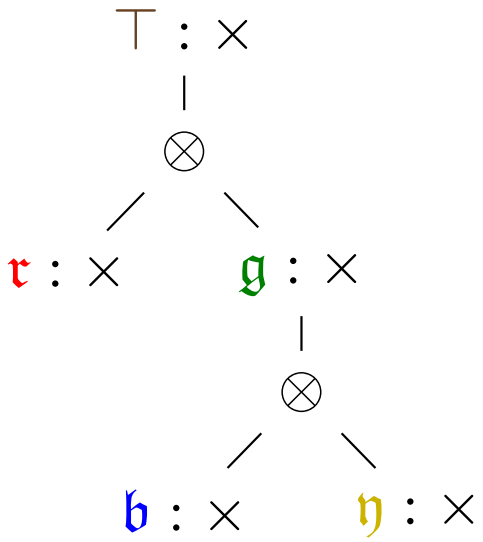
# Instead: Palettes

From earlier,

$$x : A, (y : B)(z : C, (p : P)(q : Q)), w : D$$

becomes

$$\mathfrak{r} \otimes (\mathfrak{g} \prec \mathfrak{b} \otimes \mathfrak{y}) \mid x^{\top} : A, y^{\mathfrak{r}} : B, z^{\mathfrak{g}} : C, p^{\mathfrak{b}} : P, p^{\mathfrak{y}} : Q, w^{\top} : D$$



# Tensor Types

$$\otimes\text{-FORM}\ \frac{\cdot \mid \cdot \vdash A\ \mathsf{type} \qquad \cdot \mid \cdot \vdash B\ \mathsf{type}}{\Phi \mid \Gamma \vdash A \otimes B\ \mathsf{type}}$$

We can form $A \otimes B$ for any closed types $A$ and $B$. (We will upgrade this later.)

# Tensor Types

$$\text{VAR}\ \frac{}{\Phi \mid \Gamma, x^{\top} : A, \Gamma' \vdash x : A}$$

A variable is only usable if it has the top colour.

# Tensor Types

$$\otimes\text{-INTRO}\ \frac{\Phi \text{ splits into } \Phi_L \text{ and } \Phi_R \qquad \Phi_L \mid \Gamma^{\Phi_L} \vdash a : A \qquad \Phi_R \mid \Gamma^{\Phi_R} \vdash b : B}{\Phi \mid \Gamma \vdash a \otimes b : A \otimes B}$$

Whenever we can split the palette into two pieces, use variables from one piece to prove $a$ and the other to prove $b$, then we have $a \otimes b : A \otimes B$.
Here the colours really come in handy!

# Tensor Types

$$\otimes\text{-ELIM}\ \frac{\begin{array}{c}\Phi \mid \Gamma, z^{\top} : A \otimes B \vdash C\ \mathsf{type} \\ \Phi, \mathfrak{r} \otimes \mathfrak{b} \mid \Gamma, x^{\mathfrak{r}} : A, y^{\mathfrak{b}} : B \vdash c : C[x \otimes y/z] \\ \Phi \mid \Gamma \vdash s : A \otimes B\end{array}}{\Phi \mid \Gamma \vdash \mathsf{let}\ x \otimes y = p\ \mathsf{in}\ c : C[s/z]}$$

If something holds for a *generic tensor pair* $x \otimes y$, then it holds for any particular $p : A \otimes B$.

# Eg: Symmetry

## Proposition

*For any types $A$ and $B$, there is a map* $\mathsf{swap} : A \otimes B \to B \otimes A$.

## Proof.

Suppose we have a $p : A \otimes B$. By $\otimes$-elim, we can suppose $p$ is equal to $x \otimes y$ for fresh variables $x$ and $y$. Then we have $y \otimes x : B \otimes A$. □

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ y \otimes x : A \otimes B \to B \otimes A$$

Note $\to$ is the ordinary function type here.

# Eg: Tensors and Ordinary Types

We can do whatever non-linear stuff we like with variables we have access to:

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ (x, x) \otimes y : A \otimes B \to (A \times A) \otimes B$$

# Eg: Tensors and Ordinary Types

We can do whatever non-linear stuff we like with variables we have access to:

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ (x, x) \otimes y : A \otimes B \to (A \times A) \otimes B$$

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ (x =_A x) \otimes (y =_B y) : A \otimes B \to \mathcal{U} \otimes \mathcal{U}$$

# Eg: Tensors and Ordinary Types

We can do whatever non-linear stuff we like with variables we have access to:

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ (x, x) \otimes y : A \otimes B \to (A \times A) \otimes B$$

$$\lambda p.\mathsf{let}\ x \otimes y = p\ \mathsf{in}\ (x =_A x) \otimes (y =_B y) : A \otimes B \to \mathcal{U} \otimes \mathcal{U}$$

If $f : C \otimes C \to \mathbb{N}$ we can do:

$$\lambda p.\mathsf{let}\ z \otimes w = p\ \mathsf{in}\ f(p) + f(w \otimes z) : C \otimes C \to \mathbb{N}$$

# Non-Eg: Colour Clashes

We cannot define $\Delta : A \to A \otimes A$ in general.

Given $a : A$, forming $a \otimes a : A \otimes A$ is not allowed: the two inputs to $\otimes$-intro are not well-formed in disjoint pieces of the palette.

# Non-Eg: Colour Clashes

We cannot define $\Delta : A \to A \otimes A$ in general.

Given $a : A$, forming $a \otimes a : A \otimes A$ is not allowed: the two inputs to $\otimes$-intro are not well-formed in disjoint pieces of the palette.

We cannot define $e : (A \otimes (A \to B)) \to B$ in general.

We can destruct a term of $A \otimes (A \to B)$ into $x : A$ and $f : A \to B$, but $f(x)$ is not well formed: neither variable has the top colour, so can't be used.

# Dependency?

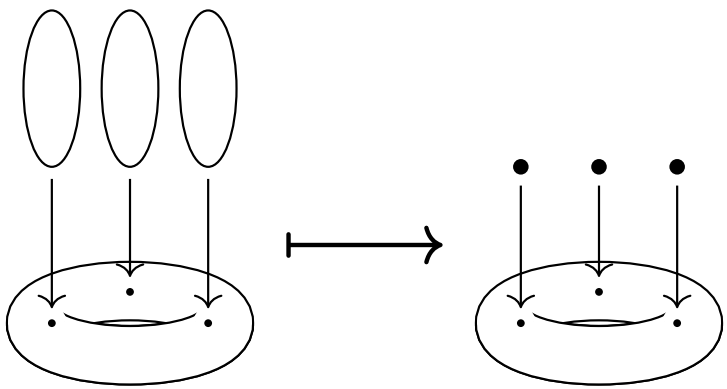
$$\otimes\text{-INTRO}\ \frac{\Phi \text{ splits into } \Phi_L \text{ and } \Phi_R \qquad \Phi_L \mid \Gamma^{\Phi_L} \vdash a : A \qquad \Phi_R \mid \Gamma^{\Phi_R} \vdash b : B}{\Phi \mid \Gamma \vdash a \otimes b : A \otimes B}$$

The syntactic issue with dependency is when we drop variables in $\otimes$-intro.

Inspecting the semantics, $(x : A) \otimes B(x)$ only makes sense when $B(x)$ only varies over the *underlying space* of $x$.

# Underlying Space

For every type $A$ there is a type $\flat A$ that deletes the spectral information.



# Marked Variables

We have to add a context version of this operation: we call it marking, and add a special 'marked variable' rule.

$$\textsc{var-marked}\ \frac{}{\Phi \mid \Gamma, x^{\mathfrak{r}} : A, \Gamma' \vdash \underline{x} : \underline{A}}$$

$$\otimes\textsc{-form}\ \frac{\cdot \mid \underline{\Gamma} \vdash A\ \mathsf{type} \qquad \cdot \mid \underline{\Gamma}, \underline{x} : A \vdash B\ \mathsf{type}}{\Phi \mid \Gamma \vdash (\underline{x} : A) \otimes B\ \mathsf{type}}$$

$\underline{x}$ can be used anywhere, even if $x^{\mathfrak{r}}$ doesn't have the top colour.

# Marked Variables

Now, when splitting the context, don't drop anything, just require it be used marked:

$$\otimes\text{-INTRO}\ \frac{\cdot \mid x : A, \underline{y} : B(\underline{x}) \vdash c : C \qquad \cdot \mid \underline{x} : A, y : B(\underline{x}) \vdash d : D}{\mathfrak{r} \otimes \mathfrak{b} \mid x : A, y : B(\underline{x}) \vdash c \otimes d : C \otimes D}$$

# Other Linear Dependent Theories

- (Schöpp and Stark 2004) is a dependent bunched theory, but has no dependency across bunches, and assumes the unit is terminal.
- (Vákár 2014) has linear type formers, but its dependent pairs/functions work differently to MLTT.
- 'LNL' type theories (Isaev 2020; Krishnaswami, Pradic and Benton 2015) separate linear types from non-linear types, so existing synthetic results can't be used.
- Quantitative Type Theories (McBride 2016; Atkey 2018) have 0-use variables that are similar our marked variables, but do not allow 'ordinary' dependence
- GRTT (Moon, Eades III and Orchard 2021) is a general framework for dependent modal theories, perhaps there is an encoding?

# 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*, pp. 56–65. DOI: 10.1145/3209108.3209189.

Barber, Andrew (1996). *Dual Intuitionistic Linear Logic*. Tech. rep. Technical Report ECS-LFCS-96-347. University of Edinburgh.

Isaev, Valery (2020). 'Indexed type theories'. In: *Mathematical Structures in Computer Science*, pp. 1–61. DOI: 10.1017/S0960129520000092.

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

Kapulkin, Chris and Peter LeFanu Lumsdaine (2012). 'The simplicial model of univalent foundations (after Voevodsky)'. In: *Journal of the European Mathematical Society*. arXiv: 1211.2851 [math.LO]. Forthcoming.

# 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: 9781450333009. 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 Comput. Sci. 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.

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

# References III


Pfenning, Frank and Rowan Davies (2001). 'A judgmental reconstruction of modal logic'. In: vol. 11, pp. 511–540. DOI: 10.1017/S0960129501003322.

Schöpp, Ulrich and Ian Stark (2004). 'A dependent type theory with names and binding'. In: *Computer Science Logic*. Vol. 3210. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 235–249. DOI: 10.1007/978-3-540-30124-0_20.

Shulman, Michael (2018). 'Brouwer's fixed-point theorem in real-cohesive homotopy type theory'. In: *Math. Structures Comput. Sci.* 28.6, pp. 856–941. ISSN: 0960-1295. DOI: 10.1017/S0960129517000147.

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

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