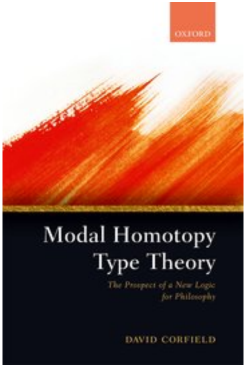
Modal homotopy type theory

David Corfield

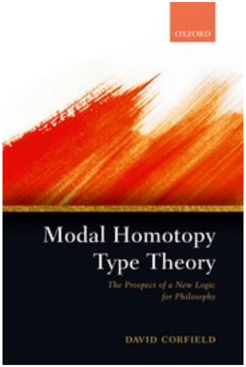
Philosophy, Kent

6 December, 2019

The prospect of a new logic for philosophy


OXFORD
Modal Homotopy Type Theory
The Prospect of a New Logic for Philosophy
DAVID CORFIELD

The prospect of a new logic for philosophy


OXFORD
Modal Homotopy Type Theory
The Prospect of a New Logic for Philosophy
DAVID CORFIELD

Available in all good bookstores February 2020.

Bertrand Russell, ‘Logic As The Essence Of Philosophy’, 1914

The old logic put thought in fetters, while the new logic gives it wings. It has, in my opinion, introduced the same kind of advance into philosophy as Galileo introduced into physics, making it possible at last to see what kinds of problems may be capable of solution, and what kinds are beyond human powers. And where a solution appears possible, the new logic provides a method which enables us to obtain results that do not merely embody personal idiosyncrasies, but must command the assent of all who are competent to form an opinion.

Philosophers today will typically use first-order logic and modal logic to ply their trade in philosophy of language and metaphysics.

This is resisted strongly by ordinary language philosophers:

> *The ideal of displaying the meaning, in particular the entailments, of sentences about events as wholly or even largely a function of structure as displayed in the canonical notation of the predicate calculus is chimerical. (Hacker 1982, p. 485)*

> *My thanks go to Dr. P. M. S. Hacker for helpful comments on the text, and to my wife, Ros, without whose patient understanding this book would not have been possible; not to mention those many writers in the tradition of Frege and Russell without whose excesses much of it would not have been necessary. (Bede Rundle 1979, p. ix)*

Can we do better?

Modal homotopy type theory

is really

Modal homotopy **dependent** type theory

Can we do better?

Modal homotopy type theory

is really

Modal homotopy **dependent** type theory

This presents us with a number of questions:

- Why types?
- Why dependent types?
- Why homotopy types?
- Why add modalities?

Why types?

- Everyday life: Natural language, jokes

Why types?

- Everyday life: Natural language, jokes
- Computing: Type checking

Why types?

- Everyday life: Natural language, jokes
- Computing: Type checking
- Metaphysics: Objects, events, properties, states of affairs, ...

Why types?

- Everyday life: Natural language, jokes
- Computing: Type checking
- Metaphysics: Objects, events, properties, states of affairs, ...
- Philosophy: Category mistakes - a visitor to Oxford upon viewing the colleges and library, reportedly inquired "But where is the University?"

A joke

A termite with a tooth-ache walks into a bar, and asks "Where's the bar tender?"

A joke

A termite with a tooth-ache walks into a bar, and asks "Where's the bar tender?"

At first we ignore much of the context, are familiar with speaking animals in jokes who address people in bars, and so take 'bar tender' as a profession, taking the question to ask for someone's location.

On finding that the joke has ended, we reappraise the context to reparse the question, and find it to be asking about the whereabouts of a subpart of an object possessing some quality.

Why *dependent* types?

- Everyday life: Natural language - Days of the month; Ways from A to B; Films starring X, Y and Z.

Why *dependent* types?

- Everyday life: Natural language - Days of the month; Ways from A to B; Films starring X, Y and Z.
- Computing: Usefully expressive, e.g., with $\mathit{Matrices}(m, n, R)$, I can ensure that we only multiply an element with one from a type of the form $\mathit{Matrices}(n, p, R)$, for $m, n, p : \mathbb{N}$ and $R : \mathit{Ring}$.

Why *dependent* types?

- Everyday life: Natural language - Days of the month; Ways from A to B; Films starring X, Y and Z.
- Computing: Usefully expressive, e.g., with $Matrices(m, n, R)$, I can ensure that we only multiply an element with one from a type of the form $Matrices(n, p, R)$, for $m, n, p : \mathbb{N}$ and $R : Ring$.
- Computing: Programs are functions and so have a type: "All errors are type errors."

Why *dependent* types?

- Everyday life: Natural language - Days of the month; Ways from A to B; Films starring X, Y and Z.
- Computing: Usefully expressive, e.g., with $\mathit{Matrices}(m, n, R)$, I can ensure that we only multiply an element with one from a type of the form $\mathit{Matrices}(n, p, R)$, for $m, n, p : \mathbb{N}$ and $R : \mathit{Ring}$.
- Computing: Programs are functions and so have a type: “All errors are type errors.”
- Mathematics/Physics: fibre bundles/gauge fields.

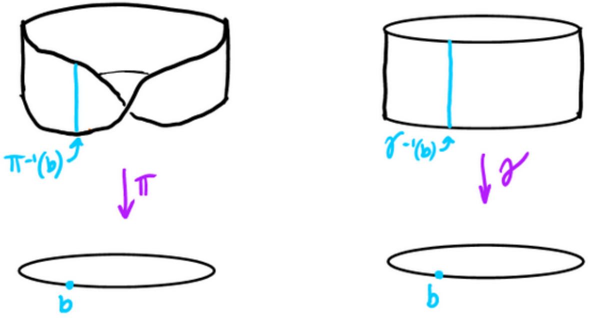
An important part of Martin-Löf type theory is the notion of a *dependent* type, denoted

$$x : A \vdash B(x) : Type.$$

Here the type $B(x)$ *depends* on an element of A, as in

- $Days(m)$ for $m : Month$
- $Players(t)$ for $t : Team$

It's helpful to have in mind the imagery of spaces fibred over other spaces:



Interpreting types as spaces, such spaces over other spaces are everywhere in mathematics and physics, fibre bundles and gauge fields.

Two central constructions we can apply to these types are **dependent sum** and **dependent product**: the total space and the sections.

Two central constructions we can apply to these types are **dependent sum** and **dependent product**: the total space and the sections.

In general we can think of this **dependent sum** as sitting ‘fibred’ above the base type A, as one might imagine the collection of league players lined up in fibres above their team name.

Two central constructions we can apply to these types are **dependent sum** and **dependent product**: the total space and the sections.

In general we can think of this **dependent sum** as sitting 'fibred' above the base type A, as one might imagine the collection of league players lined up in fibres above their team name.

Likewise an element of the **dependent product** is a choice of a player from each team, such as $Captain(t)$.

Dependent sum	Dependent product
$\sum_{x:A} B(x)$ is the collection of pairs (a, b) with $a : A$ and $b : B(a)$	$\prod_{x:A} B(x)$, is the collection of functions, f, such that $f(a) : B(a)$
When A is a set and $B(x)$ is a constant set B: The product of the sets.	When A is a set and $B(x)$ is a constant set B: The set of functions from A to B.
When A is a proposition and $B(x)$ is a constant proposition, B: The conjunction of A and B.	When A is a proposition and $B(x)$ is a constant proposition, B: The implication $A \to B$.

Dependent sum	Dependent product
$\sum_{x:A} B(x)$ is the collection of pairs (a, b) with $a : A$ and $b : B(a)$	$\prod_{x:A} B(x)$, is the collection of functions, f, such that $f(a) : B(a)$
When A is a set and $B(x)$ is a constant set B: The product of the sets.	When A is a set and $B(x)$ is a constant set B: The set of functions from A to B.
When A is a proposition and $B(x)$ is a constant proposition, B: The conjunction of A and B.	When A is a proposition and $B(x)$ is a constant proposition, B: The implication $A \to B$.

As Lawvere taught us, these are left and right adjoints.

And

Often taken as mere conjunction, $P \& Q$, as in

- Jack fell down and broke his crown, and Jill came tumbling after.
- Jack fell down and is twenty years old, and Jill is holidaying in Thailand.

Bede Rundle - there's much more to 'and'

On the one hand, I am inclined to think that the standard philosophical treatment of conjunctions like 'and', 'but' and 'although' has been grossly inadequate, no concern being shown for anything more than a narrow aspect of their use, and no investigation of that use being conducted on the right principles. On the other hand, the preoccupation with truth-conditions which has resulted in this defective approach is one towards which it is easy, and proper, to be sympathetic, at least initially. I shall begin by indicating the considerations which might invite our sympathy, and then call upon the example of 'and' to show how the approach is defective. (1983, p. 386)

- He used to lie in the sun and play cards.
- Jack fell down and broke his crown, and Jill came tumbling after.
- Pam took the key out of her bag and opened the door.
- It's raining now, and doing so heavily.

- He used to lie in the sun and play cards **then**.
- Jack fell down and broke his crown, and Jill came tumbling after **him**.
- Pam took the key out of her bag and opened the door **with it**.
- It's raining now, and doing **this raining** heavily.

Dependent sum (pair) is a good way to capture dependencies even in the case where A is a proposition

- $A : Prop, x : A \vdash B(x) : Prop$
- $\vdash \sum_{x:A} B(x) : Prop$, composed of (a, b) with $a : A, b : B(a)$.

Think of a questionnaire:

- Qu. 1: Do you have children? Yes ☐ No ☐.
  If you answered 'No' to Qu.1, go to Qu. 3.
- Qu. 2: Are any of your children aged 5 or under? Yes ☐ No ☐.
- Qu. 3: ...

There are only three possible ways to answer, not four.

In dependent type theory we can take a context to be an iterated dependent sum.

$$\Gamma \ = \ x_0 : A_0, \ x_1 : A_1(x_0), \ x_2 : A_2(x_0, x_1), \ \ldots, x_n : A_n(x_0, \ldots, x_{n-1}).$$

These are convenient for representing stories with all their dependencies.

A man walks into a bar. He's whistling a tune. A woman sits at a table in the bar. She's nursing a drink. On hearing the tune, she jumps up, knocking over the drink. She hurls the glass at him. "Is that any way to greet your husband", he says.

$x_1 : \mathit{Man}, x_2 : \mathit{Bar}, x_3 : \mathit{WalksInto}(x_1, x_2), x_4 : \mathit{Tune}, x_5 : \mathit{Whistle}(x_1, x_4), x_6 : \mathit{Woman}, x_7 : \mathit{Table}, x_8 : \mathit{Locate}(x_7, x_2), x_9 : \mathit{SitsAt}(x_6, x_7), x_{10} : \mathit{Drink}, x_{11} : \mathit{Nurse}(x_6, x_{10}), x_{12} : \mathit{Hear}(x_6, x_5), \ldots$

A man walks into a bar. He's whistling a tune. A woman sits at a table in the bar. She's nursing a drink. On hearing the tune, she jumps up, knocking over the drink. She hurls the glass at him. "Is that any way to greet your husband", he says.

$x_1 : \mathit{Man}, x_2 : \mathit{Bar}, x_3 : \mathit{WalksInto}(x_1, x_2), x_4 : \mathit{Tune}, x_5 : \mathit{Whistle}(x_1, x_4), x_6 : \mathit{Woman}, x_7 : \mathit{Table}, x_8 : \mathit{Locate}(x_7, x_2), x_9 : \mathit{SitsAt}(x_6, x_7), x_{10} : \mathit{Drink}, x_{11} : \mathit{Nurse}(x_6, x_{10}), x_{12} : \mathit{Hear}(x_6, x_5), \ldots$

There are ways to think about counterfactuals, presuppositions, ...

Why *homotopy* types?

- Mathematics: homotopification – derived geometry, higher algebra...

Why *homotopy* types?

- Mathematics: homotopification – derived geometry, higher algebra...
- Physics: Gauge equivalence, gauge-of-gauge equivalance,...

Why *homotopy* types?

- Mathematics: homotopification – derived geometry, higher algebra...
- Physics: Gauge equivalence, gauge-of-gauge equivalance,...
- Computing: rewriting and concurrency.

Why *homotopy* types?

- Mathematics: homotopification – derived geometry, higher algebra...
- Physics: Gauge equivalence, gauge-of-gauge equivalance,...
- Computing: rewriting and concurrency.
- Philosophy: identity, equivariance,...

One central choice in mathematics is the basic shape of mathematical entities:

- The set as a bag of dots, completely distinct and yet indistinguishable.

One central choice in mathematics is the basic shape of mathematical entities:

- The set as a bag of dots, completely distinct and yet indistinguishable.
- $x, y : A$, then $(x =_A y)$ is a proposition.

We ask *whether* two elements are the same, not *how* they are the same.

One central choice in mathematics is the basic shape of mathematical entities:

- The set as a bag of dots, completely distinct and yet indistinguishable.
- $x, y : A$, then $(x =_A y)$ is a proposition.

We ask *whether* two elements are the same, not *how* they are the same.

Irrespective of the way one chooses to describe sets formally, 'materially' or 'structurally', it's an astonishing idea that mathematics could rely on such a conception.

However, arising from the needs of current geometry and current physics, we find that having solely such a basic shape is a restriction. Beyond sets we need

- *Homotopy types* or *n-groupoids*: points, reversible paths between points, reversible paths between paths, ...

These may seem more complicated, but they arise in systems with *fewer* axioms.

The internal view

For any two elements of a collection we can ask whether they are the same or not.

- Where we have a collection A and $x, y : A$, we form $x =_A y$.

But then we can treat the latter as a collection and iterate.

- From $x =_A y$, and $p, q : x =_A y$, we form $p =_{(x =_A y)} q$.

Drop the ‘Uniqueness of Identity proofs’

We need not insist that any two proofs of the sameness of entities are themselves the same.

We reject the axiom that claims this is the case, or in other words we don’t insist that the following type is necessarily inhabited:

$$p =_{(x =_A y)} q \, .$$

Drop the ‘Uniqueness of Identity proofs’

We need not insist that any two proofs of the sameness of entities are themselves the same.

We reject the axiom that claims this is the case, or in other words we don’t insist that the following type is necessarily inhabited:

$$p =_{(x =_A y)} q\,.$$

This gives us a ‘simpler’ system.

We now have a hierarchy of kinds of types:

...	...
2	2-groupoid
1	groupoid
0	set
-1	mere proposition
-2	contractible type

The external view

- Gathering together all sets results in a collection which behaves nicely: a *topos*.

The external view

- Gathering together all sets results in a collection which behaves nicely: a *topos*.
- Gathering together all homotopy types results in a collection which behaves *extremely* nicely: an $(\infty, 1)$-*topos*.

The external view

- Gathering together all sets results in a collection which behaves nicely: a *topos*.
- Gathering together all homotopy types results in a collection which behaves *extremely* nicely: an $(\infty, 1)$-*topos*.

We need to tell a justificatory story running at least from Grothendieck to Lurie.

($(\infty, 1)$-toposes are a particularly nice environment for cohomology: https://ncatlab.org/nlab/show/cohomology, as I suggested yesterday.)

(homotopy type) theory **and** *homotopy (type theory)*

- *Homotopy type theory* as (homotopy type) theory is a synthetic theory of homotopy types or ∞-groupoids. It is modelled by spaces (but also by lots of other things).

(homotopy type) theory **and** *homotopy (type theory)*

- *Homotopy type theory* as (homotopy type) theory is a synthetic theory of homotopy types or ∞-groupoids. It is modelled by spaces (but also by lots of other things).
- *Homotopy type theory* as homotopy (type theory) is the internal language of ∞-toposes. It is a type theory in the logical sense, and may be implemented on a computer.

(homotopy type) theory **and** *homotopy (type theory)*

- *Homotopy type theory* as (homotopy type) theory is a synthetic theory of homotopy types or ∞-groupoids. It is modelled by spaces (but also by lots of other things).
- *Homotopy type theory* as homotopy (type theory) is the internal language of ∞-toposes. It is a type theory in the logical sense, and may be implemented on a computer.

We see *wedded together* the

- Categorical logic of William Lawvere: Adjointness in foundations.
- (Constructive) intensional type theory of Per Martin-Löf.

The bottom line is that homotopy type theory for the lower levels of the hierarchy encapsulates:

- Propositional logic
- (Typed) predicate logic
- Structural set theory

Considering the full type theory, the line between logic and mathematics has blurred – homotopy groups of the spheres, group actions,...

HoTT is a structural theory *par excellence*.

Structural inference - univalence

If A and B are equivalent types, then whatever we can establish about A may be transferred to B.

(See my *Expressing 'the structure of' in homotopy type theory*, or Ahrens and North, *Univalent foundations and the equivalence principle*.)

The

$$X : \mathit{Type} \vdash \mathit{isContr}(X) \equiv \sum_{(x : X)} \prod_{(y : X)} \mathit{Id}_X(x, y) : \mathit{Type}.$$

My proposal for ‘the’ introduction:

The

$$X : Type \vdash isContr(X) \equiv \sum_{(x : X)} \prod_{(y : X)} Id_X(x, y) : Type.$$

My proposal for ‘the’ introduction:

$$X : Type, (x, p) : isContr(X) \vdash the(X, x, p) \equiv x : X,$$

which makes sense of why we say ‘the product of two sets’ but not ‘the algebraic closure of a field’, despite all such things being isomorphic.

Why modalities?

- Everyday: invariance under modification

Why modalities?

- Everyday: invariance under modification
- Computing: staged computation, run-time code generation; trusted communications,...

Why modalities?

- Everyday: invariance under modification
- Computing: staged computation, run-time code generation; trusted communications,...
- Mathematics: synthetic axioms for continuity, smoothness

Why modalities?

- Everyday: invariance under modification
- Computing: staged computation, run-time code generation; trusted communications,...
- Mathematics: synthetic axioms for continuity, smoothness
- Philosophy: metaphysics – possible worlds, temporal logic, epistemic logic.

Why modalities?

- Everyday: invariance under modification
- Computing: staged computation, run-time code generation; trusted communications,...
- Mathematics: synthetic axioms for continuity, smoothness
- Philosophy: metaphysics – possible worlds, temporal logic, epistemic logic.
- Linguistics: pragmatics

Lawvere on quantifiers

For $\mathbf{H}$ a topos (or ∞-topos) and $f : X \to Y$ an arrow in $\mathbf{H}$, then base change induces between over-toposes:

$$(\sum_f \dashv f^* \dashv \prod_f) : \mathbf{H}/X \underset{f_*}{\overset{\overset{f_!}{\to}}{\underset{\to}{\overset{f^*}{\leftarrow}}}} \mathbf{H}/Y$$

Lawvere on quantifiers

Take a mapping

$$Owner : Dog \to Person,$$

then any property of people can be transported over to a property of dogs, e.g.,

Lawvere on quantifiers

Take a mapping

$$Owner : Dog \to Person,$$

then any property of people can be transported over to a property of dogs, e.g.,

$$\textit{Being French} \mapsto \textit{Being owned by a French person}.$$

We shouldn't expect every property of dogs will occur in this fashion.

We shouldn't expect every property of dogs will occur in this fashion.

In other words, we can't necessarily invert this mapping to send, say, 'Pug' to a property of People.

Lawvere on quantifiers

We can try...

Pug $\mapsto$ *Owning some pug* $\mapsto$???

Lawvere on quantifiers

But then

$$\textit{Pug} \mapsto \textit{Owning some pug} \mapsto \textit{Owned by someone who owns a pug}.$$

However, people may own more than one breed of dog.

Lawvere on quantifiers

How about

$$Pug \mapsto Owning\ only\ pugs \mapsto ???$$

Lawvere on quantifiers

But this leads to

$$\textit{Pug} \mapsto \textit{Owning only pugs} \mapsto \textit{Owned by someone owning only pugs}$$

But again, not all pugs are owned by single breed owners.

Lawvere on quantifiers

In some sense, these are the best approximations to an inverse (left and right adjoints). They correspond to the type theorist's dependent sum and dependent product.

Were we to take the terminal map so as to group all dogs together ($Dog \to \mathbf{1}$), then the attempts at inverses would send a property such as 'Pug' to familiar things:

'Some dog is a pug' and 'All dogs are pugs'.

Modal logic

What if we take a map $Worlds \to \mathbf{1}$?

We begin to see the modal logician's *possibly* (in some world) and *necessarily* (in all worlds) appear.

Modal logic

What if we take a map *Worlds* $\to \mathbf{1}$?

We begin to see the modal logician's *possibly* (in some world) and *necessarily* (in all worlds) appear.

Things work out well if we form the (co)monad of dependent sum (product) followed by base change, so that possibly P and necessarily P are dependent on the type *Worlds*.

This resembles this dog-owner case better if we consider just an equivalence relation on Worlds, represented by a surjection, $W \to V$. Necessarily P holds at a world if P holds at all related worlds.

These constructions applied to our pug case are:

$$Pug \mapsto Owning\ some\ pug \mapsto Owned\ by\ someone\ who\ owns\ a\ pug.$$

$$Pug \mapsto Owning\ only\ pugs \mapsto Owned\ by\ someone\ owning\ only\ pugs$$

$\bigcirc_{owner} Pug(d)$ means of a dog, d, that some co-owned dog is a pug.
$\Box_{owner} Pug(d)$ means of a dog, d, that all co-owned dogs are pugs.

We have equivalents of

- $P \to \bigcirc P$ and $\bigcirc \bigcirc P \to \bigcirc P$
- $\Box P \to P$ and $\Box P \to \Box\Box P$

Such composites will be adjoint to each other, expressing their ‘opposition’.

$$\bigcirc A(w) \to B(w) \Leftrightarrow A(w) \to \Box B(w)$$

We can now make sense of modalities applied to types that are sets, not just propositions.

- Beef is a necessary ingredient of a beef stew.
- Defeat is a possible outcome.

Much is to do with how to compare across fibres/dependent types.

Internalisation of judgements

Curry's proposal was to take $\bigcirc\phi$ as the statement "in some stronger (outer) theory, ϕ holds". As examples of such nested systems of reasoning (with two levels) he suggested Mathematics as the inner and Physics as the outer system, or Physics as the inner system and Biology as the Outer. In both examples the outer system is more encompassing than the inner system where reasoning follows a more rigid notion of truth and deduction. The modality $\bigcirc$, which Curry conceived of as a modality of possibility, is a way of reflecting the relaxed, outer notion of truth within the inner system.

(Fairtlough and Mendler, On the Logical Content of Computational Type Theory: A Solution to Curry's Problem)

Reflections of objects and morphisms across adjunctions

Adjoint

$$\begin{array}{c} A \vdash_p \mathbf{U}_f\, C \\ \hline\hline A \vdash_f C \\ \hline\hline \mathbf{F}_f\, A \vdash_q C \end{array}$$

$$\begin{tikzcd}
\mathbf{A} \arrow[d] \arrow[dr] & \mathbf{F}_f\, A \arrow[d] \\
\mathbf{U}_f\, C \arrow[d] & C \arrow[d] \\
p \arrow[r, "f", "\mathcal{M}"'] & q
\end{tikzcd}$$

Dan Licata and Felix Wellen, Synthetic Mathematics in Modal Dependent Type Theories.

Modal variants of HoTT

For modern geometry we can add another of Lawvere's discoveries: a synthetic account of cohesion via *modalities*.

(See my *Reviving the philosophy of geometry*.)

Computer science also studies modal types for permissions, etc.

Philosophical leads pointing to modal HoTT

- Logicism, constructivism, structuralism, formalism
- Computational trinitarianism
- Husserl, ...
- Metaphysics: Types, identity, modal types...
- Natural language
- Physics

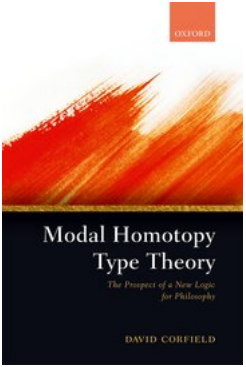
Philosophical leads pointing to modal HoTT

- Logicism, constructivism, structuralism, formalism
- Computational trinitarianism
- Husserl, ...
- Metaphysics: Types, identity, modal types...
- Natural language
- Physics

But never forget the place of category theory here.

- HoTT and $(\infty, 1)$-toposes go hand in hand.
- Modal HoTT is about functors between $(\infty, 1)$-toposes

Campaign slogan


OXFORD
Modal Homotopy Type Theory
The Prospect of a New Logic for Philosophy
DAVID CORFIELD

Modal homotopy type theory as the new logic for Philosophy (and Mathematics and Physics and Computer Science).