Type-theoretic Expressivism

David Corfield

Department of Philosophy (for the moment)
University of Kent

22 June, 2023

Starting point

My 2020 book,

- Modal homotopy type theory: The prospect of a new logic for philosophy,

provides reasons for philosophy to take up this new formalism.

HoTT as a new structural foundation for mathematics, the guidance it provides to creating new mathematics, and its role in foundational physics (I and II).

It's a stellar example of *computational trinitarianism* or *computational trilogy*.

nLab: computational trinitarianism/trilogy

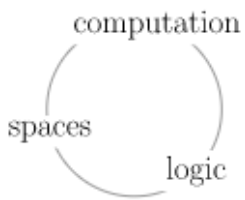
> The central dogma of computational trinitarianism holds that Logic, Languages, and Categories are but three manifestations of one divine notion of computation. There is no preferred route to enlightenment: each aspect provides insights that comprise the experience of computation in our lives.
>
> Computational trinitarianism entails that any concept arising in one aspect should have meaning from the perspective of the other two. If you arrive at an insight that has importance for logic, languages, and categories, then you may feel sure that you have elucidated an essential concept of computation–you have made an enduring scientific discovery.
>
> Bob Harper

1. Idea

A profound cross-disciplinary insight has emerged – starting in the late 1970s, with core refinements in recent years – observing that three superficially different-looking fields of mathematics,

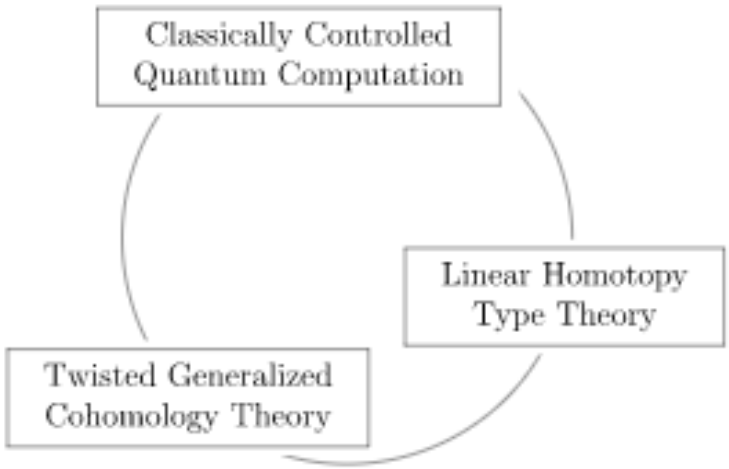
- computation/programming languages
- formal logic/type theory
- ∞-category theory/∞-topos theory (algebraic topology)

are but three different perspectives on a single underlying phenomenon at the foundations of mathematics:



nLab: computational trinitarianism/trilogy

In one recent guise (Urs Schreiber):



Something is brewing here, cf. my Thomas Kuhn, Modern Mathematics and the Dynamics of Reason.

Linking to philosophy

Can we make modal HoTT speak more directly to philosophy?

Plenty of possibilities.

- Ryle's *category mistakes*, Magidor
- Thomason on *Metaphysics made easy*
- ...
- Brandom on most things

Let's try out his logical expressivism.

Sketch of an approach

- Logical expressivism has been developed by Robert Brandom and others to understand logical vocabulary in terms of what it makes explicit about inferential practice.
- This expressivism is applied to common logical vocabulary, such as propositional implication, conjunction and negation.
- There are type theories for which such vocabulary arises by the restriction of more general constructions.
- We can give an expressivist reading of these more general constructions.
- We should have a broader understanding of the range of logic.

Motivation for dependent type theory from mathematical reasoning

Kevin Buzzard, number theorist at Imperial College London, is looking to produce computer-checked proofs of mainstream theory (schemes, perfectoid spaces, etc.) in the Lean theorem-prover.

In his talk – Is HoTT the way to do mathematics? - he asks why Lean?

Well, Lean is based on dependent type theory, and at (12:14) in the talk Kevin claims of ordinary mathematicians that

> *They use dependent types, even though they don't know they are using dependent types.*

My claim (after Aarne Ranta):

> *Mathematicians use dependent types because they are speakers of natural language. We all use dependent types.*

The case for the advantages of dependent type theory over first-order logic can be made not only for advanced reasoning such as mathematics, but also for everyday reasoning.

Dependent types

Let k be a field, V a finite-dimensional vector space over k, and f an endomorphism of V. Then define $E(V, k, f)$, the eventual image of f, as the vector space which is the intersection of all $f^n(V)$. Show $f(E) = E$.

$k : \mathit{Field}, V : \mathit{FinVect}(k), f : \mathit{Endo}(V, k) \vdash E(V, k, f) : \mathit{SubVect}(V, k)$

Let x be an author, y one of their books, and z a character from this book. The question arises as to whether z's appearance in y is autobiographical.

$x : \mathit{Author}, y : \mathit{Book}(x), z : \mathit{Character}(x, y) \vdash \mathit{Autobiographical}(x, y, z) : \mathit{Prop}$

If yes, then

$x : \mathit{Author}, y : \mathit{Book}(x), z : \mathit{Character}(x, y) \vdash a : \mathit{Autobiographical}(x, y, z)$

Judgment structure

The expressivity of our logic is determined by which judgments are allowed:

- $x : A \vdash b : B$
- $x : A, y : B, z : C \vdash d : D$
- $x : A, y : B, z : C, p : P(x), q : Q(y, z), r : R \vdash s : S(x, y, z)$
- $x : A, y : B(x), z : C(x, y) \vdash d : D(x, y, z)$
- variants with multiple consequents

(Cf. Mike Shulman's treatment of this here.)

Note that first-order reasoning doesn't exploit the full resources of the third pattern:

- $x : A, y : B, z : C, p : P(x), q : Q(y, z), r : R \vdash s : S(x, y, z)$

Usually first-order logic is untyped (or unityped), so no A, B, C:

- $\cancel{x : A, y : B, z : C}, P(x), Q(y, z), R \vdash S(x, y, z)$

The domain remains implicit, and all variables range over it.

Note also that we don't show elements of the premises and conclusion.

Brandom's account of logic: Inferentialism

- **Logical (hyper)inferentialism**: We should understand logical constants in terms of their associated rules of inference which dictate what justifies our assertion of compound propositions involving them, and what justified assertion of such compound propositions entitles us to assert further.
- This is neatly captured in logical frameworks couched in terms of introduction and elimination rules.

Introduction

$$\frac{A \vdash B}{A \to B}$$

Elimination

$$\frac{A \to B \qquad A}{B}$$

Category theory!

Introduction

$$\frac{C, A \vdash B}{C \vdash A \to B}$$

Elimination

$$\frac{A \to B \qquad A}{B}$$

$A\wedge$ as adjoint to $A \to$,

- (Intro) $Hom(A \wedge C, B) \simeq Hom(C, A \to B)$
- (Elim) $A \wedge (A \to B) \vdash B$, counit for the comonad from the adjunction.

Computational trinitarianism prevails. Logical constructions arise from a 'web of adjunctions' of elementary operations. E.g., conjunction as right adjoint to duplication: $Hom((C, C), (A, B)) \simeq Hom(C, A \wedge B)$.

Brandom's account of logic: Expressivism

- **Logical Expressivism**: "*the expressive role that distinguishes logical vocabulary is to make explicit the inferential relations that articulate the semantic contents of the concepts expressed by the use of ordinary, nonlogical vocabulary.*" (Brandom 2018, p. 70)

  For each bit of vocabulary to count as logical in the expressivist sense, one must say what feature of reasoning, to begin with, with nonlogical concepts, it expresses.(Ibid., p. 70)

(R. Brandom 2018, From logical expressivism to expressivist logic: sketch of a program and some implementations, Philosophical Issues, 28, Philosophy of Logic and Inferential Reasoning, 2018 doi:10.1111/phis.12116)

Expressivism and implication

The meaning of $\rightarrow$ is given by introduction-elimination rules.

Expressivism: It doesn't matter which propositions are involved. The logical constant $\rightarrow$ allows us to consider an inferential step explicitly. Rather than wonder whether to agree to the assertation of B on the basis of the assertion of A, I may consider the assertation of $A \rightarrow B$.

> *Intuitionistic conditionals in the broadest sense let us assert that there is a procedure for turning an argument for the premises of an inference into an argument for the conclusion. (Ibid., p. 70)*

However, from HoTT's perspective this is a restriction of type rules to propositions. Why not extend inferentialism and expressivism to the full type theory?

Note: A proposition is a type of a certain kind, and a set is a type of another kind. There are other kinds of type.

For a type A,

- $isProp(A)$ is the type that will assign to any two terms of A an element in the identity type $x =_A y$.
- $isSet(A)$ is the type that the identity types of A, $x =_A y$, are proposition types.

Dependent product/function types

Omitting mention of general contexts and omitting computational/harmony rules, the rules for dependent product (function) types in HoTT are:

Formation

$$\frac{\vdash A : \mathit{Type} \qquad x : A \vdash B(x) : \mathit{Type}}{\vdash \prod_{x : A} B(x) : \mathit{Type}}$$

Elimination

$$\frac{\vdash f : \prod_{x : A} B(x) \qquad \vdash a : A}{\vdash f(a) : B(a)}$$

Introduction

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

Propositional logic

In propositional logic there are no dependent types, and each type is a proposition.

The type formation rule becomes:

$$\frac{A : \mathit{Prop} \qquad B : \mathit{Prop}}{A \to B : \mathit{Prop}}$$

Elimination becomes

$$\frac{A \to B \qquad A}{B}$$

Introduction becomes

$$\frac{A \vdash B}{A \to B}$$

First-order logic

Reduction to untyped first-order logic. There is only one non-dependent type, the universe of discourse, which is left unnotated.

Predicates and relations are dependent propositions (so dependent types). Since this calculus doesn't signify elements of propositions, we just write a formula to indicate that it is true.

The type formation rule becomes:

$$\frac{B(x) \textit{ is a predicate}}{\forall x B(x) \textit{ is a proposition}}$$

First-order logic

Term introduction becomes

$$\frac{B(x)}{\forall x B(x)}$$

Consideration of contexts affect choices of variables.

Term elimination becomes

$$\frac{\forall x B(x)}{B(t)}$$

Category theory again

There is a *context extension* functor from $\mathcal{C} \to \mathcal{C}_{/A}$:

$$\vdash B : \mathit{Type} \mapsto x : A \vdash B : \mathit{Type}$$

this is sending a type B to

$$\begin{array}{c} A \times B \\ \downarrow \\ A \end{array}$$

Note how basic the context extension operation is – including some unrelated content into the context. I speak and then change topic. I can't wipe something from the record if subsequent items refer to it. But I can add whatever I like to the beginning. I should still assent to the deductive reasoning before me.

Category theory again

Dependent product is the right adjoint functor to this context extension functor.

Again introduction and elimination follow from this adjunction and the associated counit.

Elimination = counit

$$\frac{\vdash f : \prod_{x:\, A} B(x) \qquad \vdash a : A}{\vdash f(a) : B(a)}$$

$$(a, f) : A \times \prod_{x:\, A} B(x) \mapsto f(a) : B(a)$$

Expressivism for dependent product

Formation rule

When people consider two propositions, A and B, they should consider the proposition $A \to B$.

When people consider a predicate, $B(x)$, they should consider the proposition $\forall x B(x)$.

When people consider a type, $B(x)$, depending on a type A, they should consider the type $\prod_{x:\,A} B(x)$. E.g., states and the choice of head-of-state; cities and the choice of favourite attraction; soccer teams and their top scorers ...

Expressivism for introduction rule of dependent product

When people assent to proposition A and then because of this they assert B, then they can review this inference by naming it, indication of an element that makes $A \to B$ true.

When people consider a non-specifed entity (often indicated by the indefinite article), and judge it to have a certain quality, then they can review this judgment by wondering if it's the case that all things (of that kind) have the property. E.g., we agree that when a stranger appears, then we should fear them. This becomes a rule to consider: 'All strangers should be feared'.

Expressivism for dependent product

In the full dependent type case, when people refer to a kind, and for each element of that kind, another kind. Then if for each element of the first kind, they judge there to be an element of the dependent kind, then provide a term for that assignment.

They might give it a name. E.g., goalkeeper or captain of a football team.

Or the altitude of a plane along its flight path.

Known as a *section* in mathematics.

Same kind of story for dependent sum/pair

In propositional logic, for conjunction there are the standard FIEC rules.

Conjunction expresses co-assertion. When we're uttering propositions, and I want to know if it's from both of P and Q that I'm supposed to assent to R following.

Similarly with dependent sum. FIEC rules. Adjunction, left adjoint to context extension. Introduction rule is unit of the monad and elimination rule is from the adjunction isomorphism.

If I have a type and a dependent type, then I might consider an element of each as a pair: (Dickens, Great Expectations); (Austen, Pride and Prejudice); ...

Where are we?

- Inferentialism: meaning of dependent product and dependent sum are given by FIEC rules.
- FIEC rules are the type-theoretic version of category-theoretic adjunctions.
- These adjunctions are to context extension, a basic operation.
- Restricted use of type constructions yields logical operations.
- Unrestricted use is allowing us to make explicit aspects of our inferential practice.
- Unrestricted constructions are logical?

Still logical expressivism?

Have we strayed beyond 'logic' by taking up this broader perspective?

Recall that Brandom just needed us to "make explicit the inferential relations that articulate the semantic contents of the concepts expressed by the use of ordinary, nonlogical vocabulary".

Features of reasoning include asking for information and understanding such requests, receiving and understanding information, conducting inference, then assertion, and acceptance by others.

We request information by asking questions. Questions need to be well-formed I need to know what you're asking of me. In wondering whether your question is well-formed I may articulate to myself what I find problematic. This takes us beyond propositions.

Questions

Questions and answers can puzzle us:

- Which fruit are ripe now? Blackberries, cherries, and wolves.
- Which fruit were chordate yesterday?

I expect things to type check. I can make explicit my puzzlement using the language of types.

Questions

Aarne Ranta proposes four kinds of question with a single answer:

1. $P?$, for a proposition P
2. P or Q, which one?
3. Who, when, what X, which X, whither, whence, whose, how, how much/many/ long?: to be understood as $(Wh\ x : A)B(x)?$, for a proposition $B(x)$ depending on A.
4. Iterated question: Who read which book? Who did what to whom? This is $(Wh\ x : A)(y : B)R(x, y)?$, etc.

These may all be construed as $(Wh\ x : A)B(x)$, asking for the element of a dependent sum.

Questions

Why not just say that we ask for type listings in general?

- What are the options before us?
- Who lives here and what are their possessions?
- Which fruit are ripe now?
- How can I share these items (fairly)?

> *[The] performances Brandom dubs "assertions" have ... no recognizable subsentential semantic structure, but are internally simple, un-structured semantic "blobs". Accordingly, Brandom owes us a theory, couched in normative pragmatic terms, of subsentential expressions – names and predicates – and their meaning, and of how speakers may combine and recombine names and predicates in ever-new ways so as to produce and understand ever-new assertions and declarative sentences. (Ronald Loeffler 2018, Brandom, p.85)*

Propositions are types and are formed by type constructions. Combining and recombining names and predicates requires type discipline (cf. Ryle on type-trespasses and type-pranks), but much more is involved.

Identity type

For any type, A, and any two elements of that type, a and b, it's reasonably for me to ask $a =_A b$?

In the reduction to first-order logic, this applies solely to elements of the single non-dependent type, the domain. As a relation this is a dependent proposition, hence contains at most one element, hence we don't mention any term in it. So we can form for any two elements of the domain the proposition $s = t$.

From the HoTT perspective this derives from the general construction $a =_A b$, which applies to types themselves, $A =_{Type} B$.

Identity types have FIEC rules and arise from an adjunction (Patrick Walsh, Categorical harmony and path induction).

Expressivism for identity

Two reports from scouts as to the appearance of a stranger. Reasonable to ask whether they're speaking of the same person. I don't ask of the person reported by the first scout whether they're the same as the number 5 or the property of being blue.

If I have guests to stay, I may wonder if I have the right number of beds via the type of isomorphisms, $Guest =_{Type} Bed$.

In physics we need to go higher equivalences to allow gauge-of-gauge transformations, $g =_{gauge} g'$.

List

The List type formation

$$A : \mathit{Type} \vdash \mathit{List}(A) : \mathit{Type}$$

Comes with FIEC rules. Constructors for the empty list and to append element of A at head of a list.

Propositional truncation

If you have a type A, it's useful to have a type $||A||$. Sometimes I just care that a type is inhabited, not the identity of its elements.

I ask "Is the house occupied?" "Yes, X lives there and so does Y, I think perhaps Z too." "I don't care who lives there, it just matters whether its occupied or not."

We can apply this construction to any type. It comes with its own FIEC rules. Truncation as left adjoint to inclusion of propositions in all types.

It allows the HoTT version of the existential quantifier from dependent sum.

Many propositions we employ have used it, e.g., $||Guest =_{Type} Bed||$, an element of which means we can match guests to beds (without saying which sleeps in which).

Empty type

So far these constructions have always concerned type formation processes which depend on the assertion of some already established types.

Are there any presuppositionless types in our system?

Two basic examples are the empty type and the unit type. Consider the empty type first:

Type formation rule

$$\frac{}{\mathbf{0} : \mathit{Type}}$$

Term introduction rule: None

Empty type

Term elimination rule

$$\frac{x : \mathbf{0} \vdash \quad D(x) \ : \ \mathit{Type}}{\mathit{ind}_{(D)} : \ \prod_{x:\mathbf{0}} D(x)}$$

Computation rule: None

Why should we take $\mathbf{0}$ to be a logical construction?

FIEC rules arise from the left adjoint to the functor

$$\mathcal{C} \to *$$

For a proposition, P, negation is expressed with it: $\neg P = P \to \mathbf{0}$.

In terms of expressivism, why is it useful?

Someone asks a question: A? and you say $A = \mathbf{0}$. If A's a proposition, this says A is false. If A is set, this says A is empty.

The empty type makes explicit some part of inferential practice:

Which of our allies has come to support us? None!

Who are the occupants of that house? No-one

Is that house occupied? No

Unit type

Once we have $\mathbf{0}$, then we can generate

$$\mathbf{1} \simeq \prod_{x:\mathbf{0}} \mathbf{0} \simeq \mathbf{0} \to \mathbf{0}$$

.

FIEC rules from right adjoint to terminal functor.

But then $\mathbf{N} \simeq \mathit{List}(\mathbf{1})$. Still logical?

Inductive types

If we're happy with Identity types, List, product, coproduct, dependent sum, $\mathbf{0}$, $\mathbf{1}$, Booleans, $\mathbf{N}$, as logical, then note that they are all instances of inductive types. Why not allow them all as logical?

Then $||-||$ is a higher inductive type. If we allow all of these as logical, we let ourselves in for constructions needed for a synthetic homotopy theory.

Modality

If context extension and its adjoints are logical, then so should be the composites, $A^* \prod_{x:A} B(x)$, $A^* \sum_{x:A} B(x)$.

But these act as modal operators. E.g., dependent product followed by context extension for a typed predicate concerns the invariance of a property of an entity as the entity varies over its type, $\Box_A$.

Speaks to Brandom on the Sellars-Kant thesis (cf. Chap. 4 of my Modal HoTT book).

Variants for variational arrows: $W \to *$ (all worlds accessible), $W \to V$ (equivalence classes of accessible worlds), $T_1 \rightrightarrows T_0$ (temporal type theory), $A \leftarrow C \to B$ (Behavioral mereology).

Type universe

What of 'type' itself as logical vocabulary?

> *To ask the question To what type or category does so-and-so belong? is to ask In what sorts of true or false propositions and in what positions in them can so-and-so enter? Or, to put it semantically, it is to ask In what sorts of non-absurd sentences and in what positions in them can the expression 'so and so' enter? and, conversely, What sorts of sentences would be rendered absurd by the substitution for one of their sentence-factors of the expression 'so and so'? I adopt the word 'absurd' in preference to 'nonsensical' or 'meaningless' for the reason that both the two last words are sometimes used for noises like 'brillig' and 'abracadabra', and sometimes for collocations of words having no regular grammatical construction. Moreover, both have recently been adopted for polemical purposes in aid of a special theory. 'Absurd' has helpful associations with the reductio ad absurdum, and even its nuance of ridiculousness is useful rather than the reverse, for so many jokes are in fact type-pranks. (Ryle, Categories, Collected papers Vol. 2 p. 188)*

Type universe

Perhaps then we may say in expressivist vein that ‘type’ allows us to make explicit the issue of absurdity in our inference.

A further step sees us form a type of types, a type universe.

To consider, why we should hope a type universe to be univalent.

In summary

- Dependent type theory captures aspects of our inference.
- From the perspective of HoTT, we seem to need to move the line separating the logical from the mathematical.
- We see this from an inferentialist perspective, FIEC rules and adjunctions.
- But also from the expressivist perspective.
- Perhaps there is no such line.
- See you next year at *Logica 2024* $\simeq$ *Mathematica 2024*!