The n-Category Café

A group blog on math, physics and philosophy

January 7, 2013

From Set Theory to Type Theory

Posted by Mike Shulman

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The discussion on Tom's **recent post** [http://golem.ph.utexas.edu/category/2012/12/rethinking_set_theory.html] about ETCS, and the subsequent followup blog post of **Francois** [http://dorais.org/archives/1135] , have convinced me that it's time to write a new introductory blog post about type theory. So if you've been listening to people talk about type theory all this time without ever really understanding it—or if you're a newcomer and this is the first you've heard of type theory—this post is especially for you.

I used to be a big proponent of structural set theory, but although I still like it, over the past few years I've been converted to type theory instead. There are lots of reasons for this, the most important of which I won't say much about until the end. Instead I mostly want to argue for type theory mainly from a purely philosophical point of view, following on from some of the discussion on Tom's and Francois' posts.

I do want to emphasize again some things that sometimes get lost sight of (and which I sometimes lose sight of myself in the heat of a discussion). All foundational systems are based on valid philosophical points of view, and have their uses. The real reason for preferring one foundation over another should be practical. Type theory, of course, has lots of practical advantages, including a computational interpretation, direct categorical semantics, and especially the potential for a homotopical version; but other foundational choices have different practical advantages. But aside from practical concerns, philosophy is interesting and fun to talk about, and can be helpful when getting used to a new way of thinking when trying to learn a new foundational system—as long as we all try to keep an open mind, and focus on understanding the philosophy behind things that are unfamiliar to us, rather than insisting that they must be wrong because they're different from what we're used to.

My goal for this post is to start from material set theories (like ZFC) and structural set theories (like ETCS), and show how type theory can arise naturally out of an attempt to take the best of both worlds. At the end, I'll argue that to really resolve all the problems, we need *univalent* type theory.

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The basic objects of ZFC and ETCS are both called "sets", but they behave so differently that it can be confusing to use the same name for both. For clarity, in this post I'll call them material-sets (for ZFC) and structural-sets (for ETCS). There are a lot of differences between the two, but personally, I think one very important one is that although both kinds of sets have *elements*, their manner of "having elements" is different.

If X is a material-set, then for any other thing A, we can ask whether $A \in X$. The statement $A \in X$ is a *proposition* in the logician's sense: something which can be either true or false, which can be assumed as a hypothesis and proven as a conclusion, negated, combined with other propositions using conjunction, disjunction, and implication, quantified over, and so on. (In ZFC, the only things that exist are material-sets, so that A in the proposition "$A \in X$" must itself be a material-set. However, some material-set theories include "atoms" which are not sets, but which can be elements of material-sets.) (Also, most material-set theories are *extensional* in that a material-set X is uniquely determined by knowing for which A we have $A \in X$. This is a natural assumption if $\in$ is the only available notion.)

By contrast, if X is a structural-set, then there are some things which *by their very nature* are elements of X. (In ETCS, those things are the functions with domain 1 and codomain X.) If a thing is not *given* to you as an element of X, then it isn't an element of X, and no thing can be given to you as an element of more than one structural-set.

The statement $A \in X$ is not something that you would ever think about *proving* about two pre-existing things A and X, or assuming as a hypothesis except at the same time as you introduce A as a new variable (as a way of saying "this is the sort of thing that A is"). That is, to be maximally faithful to its usage in mathematics, $A \in X$ should not be considered as a *proposition*.

To illustrate the difference, consider a statement like "for all $x \in \mathbb{R}$, $x^2 \geq 0$". If $\mathbb{R}$ is a material-set, then this statement may be (and generally is) interpreted as a shorthand for "for all things x, if $x \in \mathbb{R}$, then $x^2 \geq 0$". But if $\mathbb{R}$ is a structural-set, then the quantifier "for all $x \in \mathbb{R}$" is an atomic operation of logic: since "$x \in \mathbb{R}$" is not a proposition, it cannot be the hypothesis of an if-then statement.

Personally, I think this aspect of structural-set theory matches mathematical practice more closely. When I say "for all $x \in \mathbb{R}$, $x^2 \geq 0$", I regard it as a *property of every real number* that its square is nonnegative. I don't regard it as a property of *every thing in mathematics* that *if* it happens to be a real number, then its square is nonnegative. Under the material-set theory reading, one particular instance of "for all $x \in \mathbb{R}$, $x^2 \geq 0$" is the statement "if the monster simple group happens to be a real number, then its square is nonnegative". I wouldn't expect most mathematicians to naturally regard that statement as part of the content of "for all $x \in \mathbb{R}$, $x^2 \geq 0$".

On the other hand, *sometimes* in mathematics, it does make sense to regard "$A \in X$" as a proposition. For instance, if L is the set of complex numbers with real part $\frac{1}{2}$, then a lot of people would really like to prove that "for all $z \in \mathbb{C}$, if $\zeta(z) = 0$ and z is not a negative even integer, then $z \in L$". Note the difference between the two uses of " $\in$ " in this statement. The quantifier "for all $z \in \mathbb{C}$" is naturally treated as atomic, with z being *given as* a complex number, rather than as an arbitrary thing with the additional hypothesis that it happens to be a complex number. On the other hand, the statement "$z \in L$" is a proposition, which might be true or false for any particular z, and is susceptible of proof or disproof (as $z \in \mathbb{C}$ is not). Thus, in this statement we see $\mathbb{C}$ behaving like a structural-set, while L behaves like a material-set.

The crucial observation here is that L is a *subset* of $\mathbb{C}$. Thus, since z is *given as* an element of the structural-set $\mathbb{C}$, we can *ask* whether it happens to belong to this subset L. This situation is generic, and it gives us a reasonable way to combine structural-sets and material-sets in a single theory. Namely,

> Material-sets are subsets of structural-sets.

Thus, an expression like $A \in X$ has two different meanings depending on what X is. If X is a structural-set, then "$A \in X$" is not a proposition; instead it is a way of describing how A is given. On the other hand, if A is given as an element of some structural-set Z, of which X is a subset, then "$A \in X$" *is* a proposition. We might refer to these two meanings of $\in$ as the "quantifier-bounding membership" and the "propositional membership". In my experience, this way of thinking captures the vast majority of usages of sets in mathematics.

From this perspective, we can see that some of the awkwardnesses of ZFC and ETCS arise from forcing one kind of set into the mold of the other. In ZFC, there are no structural-sets, forcing us to interpret the structural membership expression in terms of the material one, as above. On the other hand, in ETCS there are *only* structural-sets, forcing us to interpret the material membership proposition in terms of these. (Specifically, we define a "subset" of a structural-set Z to be a structural-set X equipped with an injective function $i : X \to Z$. Then if $a : 1 \to Z$ is an element of Z, we define "$a \in X$" to mean "there exists an element $a' : 1 \to X$ such that $i \circ a' = a$".)

Rather than try to resolve these inelegancies directly, however, I want to argue that they are merely symptoms of a deeper problem afflicting both material and structural set theories: an impoverishment of mathematical objects. In ZFC, everything is a material-set—even the elements of other material-sets. In ETCS, everything is either a structural-set or a function between such (with elements of structural-sets being particular functions). Other set theories have different "basic objects" — for instance, we can modify ZFC to contain "atoms" as well as material-sets, while in **SEAR** [http://ncatlab.org/nlab/show/SEAR] the basic objects are structural-sets, elements of structural-sets, and relations between them. But in all cases, the list is very short, forcing us to "encode" other mathematical objects in terms of these. Sometimes the encoding is more natural materially, sometimes it is more natural structurally, and sometimes it is unnatural in either one.

On one hand, I think power sets seem more natural materially. In ZFC, the elements of the power set $\mathscr{P}(\mathbb{N})$ *are themselves* subsets of $\mathbb{N}$. As such, we can ask, for any $X \in \mathscr{P}(\mathbb{N})$ and $n \in \mathbb{N}$, whether $n \in X$. (Note that the first two $\in$'s are quantifier-bounding, while the third is a proposition). By contrast, in ETCS, the elements of $\mathscr{P}(\mathbb{N})$ are not themselves, intrinsically, subsets of $\mathbb{N}$. Instead they are "codes" for subsets of $\mathbb{N}$, and there is a relation between $\mathbb{N}$ and $\mathscr{P}(\mathbb{N})$ called "membership" which has the expected properties.

On the other hand, I think the natural numbers seem more natural structurally. In ETCS, the natural numbers object is a structural-set $\mathbb{N}$ equipped with an element 0 and a function $s : \mathbb{N} \to \mathbb{N}$ satisfying the laws of induction and recursion. An element of $\mathbb{N}$ is "a natural number" and has no structure other than that. By contrast, in ZFC, we generally define the natural numbers to be some particular material-set, such as $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \dots\}$ or $\{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots\}$. In this case we can construct all the usual operations, but we also have other unwanted results such as $0 \in 1$ that we have to consciously "abstract away from".

Finally, on the **gripping hand** [http://www.urbandictionary.com/define.php?term=On+The+Gripping+Hand] , I don't think that function sets are particularly natural in either case. In ZFC, we have to define ordered pairs in some particular way, such as $(a, b) = \{\{a\}, \{a, b\}\}$, and define a function to be a certain kind of set of ordered pairs. To obtain the set B^A of functions between two specified sets A to B, we usually also want to annotate these functions with A and B. Thus we have a set B^A, and we can define a notion of "application" of such a function to an element of A, but we also have other unwanted properties. By contrast, in ETCS, the set B^A is characterized by a universal property. Its elements are not, literally, functions from A to B (which are basic things in the theory of ETCS), but only "codes" for them, which get identified with functions via an "evaluation" function $\mathrm{ev} : B^A \times A \to B$.

We can try to apply various band-aid fixes for these problems. For instance, we might augment ZFC with atoms, and define the natural numbers to be a set of atoms. We might similarly augment it with a basic "ordered pair" operation, or with a basic notion of "function" in addition to "set". But it would be better to have a principled solution which solves all such problems at once. Thus, we ought to seek a foundational system which

1. contains structural-sets and their elements as basic things, but also subsets of these which behave like material-sets;
2. contains both the structural quantifier-bounding membership and the material membership proposition; and
3. allows elements of structural-sets to have *some* internal structure, but doesn't force them to always have a particular *kind* of internal structure. The elements of a structural-set should have different structure depending on what that structural-set is: they may be material-sets, or functions, or ordered pairs, or natural numbers.

In fact, such a foundational system already exists: it is called Martin-Löf dependent type theory (MLTT). The language we use in MLTT is a bit different from that of set theory, but at the level of informal mathematics, the differences really amount to nothing more than changes in terminology and notation. Instead of "structural-sets" we speak of *types*. Sometimes we call their elements *terms*. And the quantifier-bounding membership is written as $a : Z$ rather than $a \in Z$ (here Z, of course, is a type—that is, a structural-set—while a is being given as an element of it).

In MLTT, we have many different ways to construct (or "form") types, and each type-forming operation comes with rules specifying what the elements of that type are and how they behave. For instance, the rule for forming cartesian product types essentially says that the elements of $A \times B$ are, by definition, pairs (a, b) where $a : A$ and $b : B$. The pair-forming operation is primitive and not defined in terms of any other structure. This is in contrast to ZFC, where (a, b) must be defined in terms of set-membership, and ETCS, where (a, b) is only characterized as such relative to a choice of a cartesian product diagram for A and B.

Similarly, the rule for forming function types essentially says that the elements of B^A are, by definition, rules associating to every element of A an element of B. Formally, by this we mean an expression denoting an element of B which contains a specified free variable of type A, such as "$x^2 + 1$" where $x : \mathbb{R}$. The resulting element of B^A is often denoted by a λ which binds the appropriate variable, such as "$\lambda x.\ x^2 + 1$" for the element of $\mathbb{R}^\mathbb{R}$ defined by the expression "$x^2 + 1$". There is a basic operation of *application* which associates to every $f : B^A$ and every $a : A$ an element $f(a) : B$, with the expected properties, e.g. $(\lambda x.\ x^2 + 1)(3) = 10$. This is in contrast to ZFC, where functions must be defined as sets of ordered pairs, and ETCS, where elements of B^A are "codes" for functions rather than literally functions themselves.

There are several other type-forming operations, but of particular note is the former for subsets. (Type theorists bear with me for a moment; this paragraph is a tad unusual.) Analogously to function-types, for any type A we may have a type $\mathscr{P}(A)$ whose elements are expressions of the form $\{x\colon A \mid \phi(x)\}$, where $\phi(x)$ is a property involving a free variable x of type A. And we have a relation $\in$ between elements of A and subsets, with the expected properties, e.g. $a \in \{x\colon A \mid \phi(x)\} \;\Leftrightarrow\; \phi(a)$. This relation $\in$ is a basic aspect of the type $\mathscr{P}(A)$ and not defined in terms of anything else, just as function application is a basic aspect of the type B^A. Thus, the elements of $\mathscr{P}(A)$ *are* literally the subsets of A, and these material-sets coexist in just the right way with the structural-sets (i.e. types) of the ambient theory.

In fact, usually in type theory $\mathscr{P}(A)$ is defined to be the function type Ω^A, where Ω is the "type of propositions" or "truth values", with $\{x\colon A \mid \phi(x)\}$ defined to mean $\lambda x.\ \phi(x)$. Then the membership relation $\in$ reduces to function application, yielding exactly the correct rules. This amounts to defining subsets to be their characteristic functions, which may seem very natural to a category theorist. But if you don't like it, you can ignore it: there's no problem with a type theory that defines $\mathscr{P}(A)$ directly as I described above.

One last type-forming operation of note is a *universe type*, whose elements are other types. By combining this with dependent sums, we can build types whose elements are *structured types*. For instance, there is a type whose elements are pairs (A, s) where A is a type and $s\colon A \to A$ is an endofunction. The ability to build sets of this sort in ZFC is, I think, what Francois **calls** [http://dorais.org/archives/1135#comment-751567992] "using sets as data structures". In ETCS the only way to talk about a structured set is to do the "pairing" at a meta-level, regarding a quantifier such as "for all groups" as abbreviating "for all structural-sets G and functions $m\colon G \times G \to G$ such that the axioms of a group hold". In type theory, we can avoid this infelicity, but we don't have to use sets as data structures either: we can use *data structures* as data structures.

At this point, however, you may be feeling that type theory sounds very complicated. Lots of different ways to form types, each with their own rules for forming elements? Where is the simplicity and intuitiveness that we expect of a foundational theory?

Well, first of all, there are not that many type-forming rules. In fact, if we allow "rule schemas" (in the same way that ZFC has "axiom schemas" such as separation and replacement), then there are only three: dependent products, inductive types, and universes. (One might add coinductive types as a fourth.) Certainly that compares reasonably well with the half-dozen or dozen axioms (or axiom schemas) of ZFC and ETCS.

Secondly, every type-forming rule conforms to a well-understood pattern: there is a part for *formation* (making a new type), a part for *introduction* (how to construct elements of that type), a part for *elimination* (how to extract information from an arbitrary element of that type), and a part for *computation* (how introduction and elimination interact). This uniformity makes the meta-theoretic analysis of type theory particularly simple, allowing proofs of "consistency by evaluation". It makes type theory into essentially a programming language, at the same time as a foundational system, allowing the easy verification and construction of proofs by computers. And it governs a principled approach to the construction of new models, and matches very closely the category-theoretic notions of left and right universal properties. (Some, but not all, of these advantages are shared by structural-set-theoretic axioms that describe universal properties in the category of sets.)

Even with all of this said, however, you may still feel that a *conceptual* simplicity is lacking. In ZFC, we have only the concepts of *set* and of *membership*. In ETCS, we have only the concepts of *set* and *function*, along with identities and composition. The axioms, you may say, only stipulate how these basic concepts behave.

In fact, one may argue that a foundational system *should* be about encoding more complicated concepts into simpler ones, even if the encoding turns out to be somewhat unnatural. For instance, the fewer basic concepts we have to deal with, the easier it is to prove consistency results and construct new models from old ones.

However, I claim that the conceptual simplicity of set theories vis-a-vis type theories is an illusion, because both material-set theories and structural-set theories are built on top of the edifice of first-order logic. To introduce them formally, therefore, we must first describe the rules for forming first-order formulas, with connectives such as conjunction, disjunction, and implication, and quantifiers such as "for all" and "there exists", and also the rules governing inference that tell us how to construct proofs of formulas involving connectives and quantifiers. In other words, ZFC and ETCS are not just theories of sets; they are theories of sets *and propositions*. The interplay between sets and propositions is most obvious in the axiom schemas such as replacement, which introduce an arbitrary first-order formula. It is also evident when constructing a new model of set theory, in which case one must first specify how all logical formulas are to be interpreted.

By contrast, type theory is *not* built on top of first-order logic; it does not require the imposing superstructure of connectives, quantifiers, and inference to be built up before we start to state axioms. Of course, type theory *has* first-order logic, which is a necessity for doing mathematics. But first-order logic in type theory is just a special case of the type-forming rules. A proposition is merely a certain type; to prove it is to exhibit an element of that type. When applied to types that are propositions, the type-forming operations of cartesian product, disjoint union, and function types reduce to the logical connectives of conjunction, disjunction, and implication; the quantifiers arise similarly from dependent sums and products. Thus, type theory is not an alternative to set theory built on the same "sub-foundations"; instead it has re-excavated those sub-foundations and incorporated them into the foundational theory itself. So not only is it more faithful to mathematical practice than either kind of set theory, it is literally simpler as well.

The problem of encoding, however, has not gone away entirely. In type theory, we can see it as another aspect of *modularity* in software engineering. For instance, the usual definition of the natural numbers in type theory is as an inductive type generated by zero and successor. However, when we regard type theory as a programming language, this definition gives rise to a "unary" representation, which is computationally inefficient. Instead, we could define a "binary" representation, involving as basic the operations of "doubling" and "doubling and adding one". Mathematically, these two definitions should be "equivalent", but they are not literally identical, in the same way that $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}\dots\}$ and $\{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots\}$ are not.

The solution to this problem is the *univalence axiom*, which can be concisely described as saying that "isomorphic structures are equal". More specifically, it says that if A and B are types, then proving that $A = B$ is the same as specifying an *isomorphism* between A and B. This implies that analogously, *structured* types, such as natural numbers objects, groups, rings, fields, etc., are equal if they are isomorphic by an isomorphism preserving the structure. In particular, the unary and binary definitions of the natural numbers can be shown to be isomorphic (as sets, or as monoids, or semirings, or whatever structure we are interested in), and therefore *equal*. Thus, anything we say about one of them is automatically equally true of the other. More generally, the "type of natural numbers objects" can be shown to contain exactly one element, so that we can speak freely of "the natural numbers" without any abuse of language—the natural numbers are *literally unique*.

In set theoretic foundations, of course, we could already prove that any two natural numbers objects were *isomorphic*. In material-set theory, all we can conclude from that is that any "isomorphism-invariant property" of one natural numbers object carries over to another. Then we have to do work (or, as is usually the case, leave work to the reader) proving that any particular property we care about is isomorphism-invariant. The situation in structural-set theory is somewhat better: if a structural-set theory is presented carefully enough, then we can only say isomorphism-invariant things. Both of these approaches, however, eventually lead to dependent type theory anyway: it is the way to characterize the isomorphism-invariant properties, and the careful way to present structural set theories. Moreover, in univalent type theory we can prove the statement "all properties are invariant under isomorphism" directly, rather than only observing it as a meta-property of the theory.

Univalence is made possible precisely by the incorporation of first-order logic into type theory at a basic level. Since the proposition $A = B$ is, like any proposition, a *type*, which is proven by giving an element of it, its elements can be things with appropriate structure. When A and B are types, therefore (that is, elements of a universe type), we can require that the elements of $A = B$ are isomorphisms from A to B. This is unworkable in classical first-order logic, where propositions and proofs exist at a different level than sets, and in particular proofs cannot contain *information*, such as which isomorphism between A and B was used to prove $A = B$. And retaining this information is essential, because to say that a property or structure is "isomorphism-invariant" involves transporting it *along a particular isomorphism*.

Univalence does require some getting used to, since as you can see, "equality" in univalent type theory behaves a bit differently from what you are probably used to. Fortunately, for most kinds of concete mathematics, this funny behavior of equality doesn't arise. The best way to think about this is that "types" are a *more general* kind of thing than structural-sets. Some types do behave just like structural-sets; in univalent type theory we call them "h-sets". However, other types, such as the type *of* h-sets, or the type of algebraic structures of some sort, have "higher h-level", and that's when equality starts to behave more unusually. All we've done is to expand the universe of mathematics to contain new things; no *familiar* object behaves any differently than it used to.

Posted at January 7, 2013 6:21 AM UTC

TrackBack URL for this Entry: https://golem.ph.utexas.edu/cgi-bin/MT-3.0/dxy-tb.fcgi/2587

Some Related Entries

Rethinking Set Theory — *Dec 18, 2012*
Universe Polymorphism and Typical Ambiguity — *Dec 09, 2012*
Freedom From Logic — *Nov 09, 2012*
The Ax-Grothendieck Theorem According to Category Theory — *Sep 10, 2012*
The Gamification of Higher Category Theory — *Jun 18, 2012*
Logic as Invariant-Theory — *Jan 13, 2012*
Propositions as Some Types and Algebraic Nonalgebraicity — *Jan 12, 2012*
Weak Systems of Arithmetic — *Oct 11, 2011*

151 Comments & 1 Trackback

view chronologically

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Mike wrote:

> For instance, if L is the set of complex numbers with real part $\frac{1}{2}$, then a lot of people would really like to prove that "for all $z \in \mathbb{C}$, if $\zeta(z) = 0$, then $z \in L$".

I hope you email these poor people and let them know that $\zeta(-2) = 0$ [http://en.wikipedia.org/wiki/Riemann_zeta_function#Zeros.2C_the_critical_line.2C_and_the_Riemann_hypothesis] .

Posted by: John Baez on January 7, 2013 7:35 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Fixed.

Posted by: Mike Shulman on January 7, 2013 8:10 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I know that this is kind of late, but maybe you should fix it even more to refer directly to the set N of negative integers. This would show explicitly how proposition-membership may be negated and used in a hypothesis.

Posted by: Toby Bartels on May 11, 2013 11:53 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Sign me up for type theory!

Rather than your "gripping hand", which, following your **link** [http://www.urbandictionary.com/define.php?term=On+The+Gripping+Hand] , suggests a forceful overriding of two alternatives by a third, how about the language of dialectics, say, the *aufhebung* ('sublation' or 'overcoming') of the Hegelian **variety** [http://en.wikipedia.org/wiki/Dialectic#Hegelian_dialectic] , where aspects of the initial proposal and of its negation are retained in the product?

Posted by: David Corfield on January 7, 2013 10:09 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Whatever you like. I'm not about to spend time arguing about language at that meta of a level!

Posted by: Mike Shulman on January 7, 2013 6:16 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

"this definition gives rise to a "unary" representation, which is computationally inefficient."

Why is a unary representation computationally inefficient?

"the natural numbers are literally unique."

How does that affect the concept of non-standard models of arithmetic?

"the proposition A=B is, like any proposition, a type, which is proven by giving an element of it"

This sounds weird but also sounds like it is one of the main things to know about type theory. Can you give an example of a proposition and an element and explain why giving an element is the same thing as proving the proposition?

Posted by: Grapr on January 7, 2013 12:23 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> Why is a unary representation computationally inefficient?

Heck, it's *storage* inefficient. Would you rather remember 3^{630} as 1000 bits, or as 3^{630} hatch-marks? (You know… provided that "3^{630}" is not available…)

> How does that affect nonstandard models of arithmetic?

Not at all: a model of arithmetic is a set with various maps among a few of its products, with simple properties; a natural numbers object is a set with an element and a map with a **universal property** w.r.t. *all sets with an element and a map*. Some would call the former a first-order notion and the latter a second-order notion. I'd phrase it differently: being a model of arithmetic is a purely internal matter, while being a natural numbers object is ambient/external.

Posted by: Jesse McKeown on January 7, 2013 3:00 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> Heck, it's *storage* inefficient.

Indeed. And no algorithm can run faster than the stored data that it needs to access.

> being a model of arithmetic is a purely internal matter, while being a natural numbers object is ambient/external.

That seems a reasonable way to say it. But there's also another sort of "nonstandard model" which may be worth mentioning in case someone confuses it: the natural numbers are unique within any model of set theory or type theory, but if you pick a different model of the ambient theory, then it will contain a different natural numbers object, which may or may not be the same as (or even at all comparable to) the one in your original model.

> Can you give an example of a proposition and an element and explain why giving an element is the same thing as proving the proposition?

Sure, here's a simple one: suppose P and Q are propositions. Then the proposition "P implies Q" is identified with the type of functions from P to Q, which type theorists usually write as "$P \to Q$". An element of this type is an operation which transforms elements of P (that is, proofs of P) to elements of Q (that is, proofs of Q). Thus, it is essentially a proof of Q *assuming* P as a hypothesis, which is the intended meaning of an implication.

Posted by: Mike Shulman on January 7, 2013 6:30 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Thanks a lot for this wonderful post! It really seems to me now that the univalent type theory could actually become the third major foundation of mathematics (the first one being ZFC and the second one being ETCS).

Are there any references that explain UTT in more depth, in particular explain the concept of univalence with more details (the description in the post is rather vague), as well as how to translate from ZFC/ETCS to UTT, for somebody who already has a working knowledge of ZFC and ETCS?

Posted by: Dmitri Pavlov on January 7, 2013 1:34 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

You can try the **references** [http://ncatlab.org/nlab/show/homotopy+type+theory#References] at nLab's **homotopy type theory** [http://ncatlab.org/nlab/show/homotopy+type+theory] page.

Posted by: David Corfield on January 7, 2013 2:17 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

And also the **links** [http://homotopytypetheory.org/links/] and **references** [http://homotopytypetheory.org/references/] pages at the **homotopy type theory web page** [http://homotopytypetheory.org/] .

Posted by: Mike Shulman on January 7, 2013 6:31 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Since I endlessly argued with you on the other thread, let me say that I like this post a great deal.

I do find Martin-Lof type theory hard to learn, though. Though recently I was able to make sense of System F from the point of view of "How would I implement this in Common Lisp?" Maybe that would work here.

Posted by: Walt on January 7, 2013 5:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I agree, it's not at all easy to learn. One thing I found hard to learn is how equality is treated in intensional type theory. The usual formal layout in terms of formation-introduction-elimination-computation can look like a lot of opaque symbolic gobbledygook.

But a basic underlying idea, that equality types can be considered as inductive types, came to me as quite a revelation. You can get an idea of how this goes from these **MathCamp notes** [http://www.math.ucsd.edu/~mshulman/papers/induction.pdf] by Mike.

Mathematics is strange, in that even after decades of work at a high level, even the most basic concepts can be understood in a new light. The idea of mathematical equality seems so basic, you know, almost the first thing you should learn. And yet after learning about induction on equality, I felt I had only just begun to understand it. Very strange indeed!

Posted by: Todd Trimble on January 7, 2013 6:47 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I finally had a chance to read Mike's slides (which are really great). Thanks for the link.

It seems pretty close to the second-order definition of equality, where $x = y$ if for all predicates, $P(x) \leftrightarrow P(y)$. I guess you can formulate that definition in dependent type theory, but this definition is aesthetically efficient.

Posted by: Walt on January 28, 2013 10:12 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

For future readers of this thread: Mike's MathCamp slides on "Induction on Equality" can now be found **here** [http://www.math.ias.edu/~mshulman/papers/induction.pdf] .

Posted by: Stuart Presnell on May 23, 2013 7:54 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

For those of us living even further in the future, the slides can be found here: https://home.sandiego.edu/~shulman/papers/induction.pdf. (In the (likely?) event that that link is no longer accessible, both this URL and the one in the preceding comment (though not the original) can be found in functional form in the Internet Archive.)

(Is it just me, or are educational institutions uniquely and utterly *terrible* at setting up redirects? If I find a link to a page on a university website posted at least a few years ago, there's like a 15% chance it actually leads somewhere. You'd think that universities of all institutions would be capable of setting up a redirect page!)

(ps: why does this comment box keep adding "[object Object]"?)

Posted by: gbd_628 on February 5, 2019 1:02 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Utterly terrible? Yes. Uniquely so? Absolutely not.

Posted by: Mark Meckes on February 5, 2019 1:18 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Could say a few words about the word choice "univalent"? I usually only think of that word in the context of vertices of graphs, and I can't really grok what it's meant to be communicating in this context.

Posted by: Noah Snyder on January 7, 2013 5:12 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The *origin* of the word "univalent" is a bit murky; I think it comes by way of **complex analysis** [http://en.wikipedia.org/wiki/Univalent_function] and (at least according to Voevodsky) is originally attributable to a mistranslation between English and Russian.

However, I think there is a good *ex post facto* explanation. You can think of "uni-valent" as meaning "single-valued" (this is not so far from its meaning in complex-analysis). Univalence means that every statement or operation respects isomorphism (because isomorphism is equality). In other words, no operation can give *more than one different result* when applied to inputs which are *the same* in the appropriate sense.

Posted by: Mike Shulman on January 7, 2013 6:37 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Thanks for writing this, Mike. I am also of the opinion that univalent type theory is objectively a better foundation than set theory (in any form). However, there is an issue with accessibility which is still unresolved. I saw that the HoTT crew is starting to look into this in an **earlier post** [http://golem.ph.utexas.edu/category/2012/11/freedom_from_logic.html] . I hope that someday there will be an article in the same spirit as **Tom's article** [http://golem.ph.utexas.edu/category/2012/12/rethinking_set_theory.html] for type theory instead of ETCS.

Posted by: François G. Dorais on January 7, 2013 7:28 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Wow, what a vote of support! Yes, we're hoping that one result of the IAS year will be something introductory and elementary to show people how to think and do mathematics in univalent foundations. What we have in mind will actually be much longer and more comprehensive than Tom's paper, though it will start out with analogous sorts of remarks (perhaps even some of what I've written in this post will make it in).

Posted by: Mike Shulman on January 7, 2013 7:37 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Mike, the explanation of function types in your post confuses me.

You say that in ETCS, the function set B^A is characterized by a universal property. This would seem to admit realist interpretations (for those who would like to entertain such interpretations) by which B^A consists of an element representing each of "all the functions there are." In particular the set of functions could be uncountable.

You say that in type theory, the elements of the function set B^A are, "by definition," rules, presented syntactically as expressions denoting elements of B containing a specified free variable of type A. The most straightforward way I can read this suggests that the set of functions would be countable, and so it would not satisfy those who wanted to entertain realist interpretations. Was it meant to be this restrictive, or am I reading you wrong?

Posted by: Kevin Watkins on January 7, 2013 10:10 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

No, certainly there is no countability implication! I'm not sure exactly what's suggesting countability to you, but maybe the following will help. In set theory, a function from A to B is defined to be a subset $F \subseteq A \times B$ with the property that for each $a \in A$, there is exactly one $b \in B$ such that $(a, b) \in F$. In type theory, a function is not by definition such a subset, but from any such subset we can build a function according to the rule "to each $a : A$, associate the unique $b : B$ such that $(a, b) \in F$." So there are just as many functions in type theory as there are in set theory.

Posted by: Mike Shulman on January 8, 2013 4:22 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Kevin got countability from the idea that an element of B^A is a finite-length expression in a finite set of symbols. That was his interpretation of this remark:

> Similarly, the rule for forming function types essentially says that the elements of B^A are, by definition, rules associating to every element of A an element of B. Formally, by this we mean an expression denoting an element of B which contains a specified free variable of type A, such as "$x^2 + 1$" where $x : \mathbb{R}$.

There are countably many things like this.

Posted by: John Baez on January 8, 2013 5:09 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Ah, okay! The point is that those expressions can also contain variables belonging to some other type, i.e. they can be “parametrized”. There are certainly only a countable number of elements of B^A that we can *define*, in the empty context (i.e. that contain no free variables). That’s equally true in set theory, for the same reasons. But, for instance, in the context containing a free variable $f : \mathbb{N} \to 2^{\mathbb{N}}$, by the usual diagonalization argument we can write down a function $(\lambda x.\, \neg f(x)(x)) : \mathbb{N} \to 2$ that is not in the image of f. Thus, it follows that $2^{\mathbb{N}}$ is uncountable, just as in set theory.

Posted by: Mike Shulman on January 8, 2013 5:21 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

But if you consider 2 to the N to consist only of the functions we can define, you still get the same diagonalization, but suddenly we choose not to interpret it as meaning 2 to the N is uncountable. Perhaps we’re dealing with a logical/structural issue, rather than a size issue. Perhaps the idea of one-to-one mappings defining equal size is just another thing we seem to understand from finite examples, but which isn’t the right concept for infinite situations.

Posted by: Hendrik Boom on January 10, 2013 2:51 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I wonder whether you are confusing “uncountable *within* the theory” with “uncountable in the metatheory”? Even if the only actual elements of 2^N were definable, it could still be true within the theory that 2^N is uncountable. Cf. **Skolem’s paradox** [http://en.wikipedia.org/wiki/Skolem%27s_paradox] .

Posted by: Mike Shulman on January 10, 2013 3:45 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

… and, for an even more striking perspective, see the work by Hamkins, Linetsky and Reitz on **pointwise definable models** [http://jdh.hamkins.org/pointwisedefinablemodelsofsettheory/] .

Posted by: François G. Dorais on January 10, 2013 5:50 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Let me try again, this time with pairs instead of functions.

You contrast pairs in MLTT with pairs in ZFC thus:

> For instance, the rule for forming cartesian product types essentially says that the elements of $A \times B$ are, by definition, pairs (a, b) where $a : A$ and $b : B$. The pair-forming operation is primitive and not defined in terms of any other structure. This is in contrast to ZFC, where (a, b) must be defined in terms of set-membership.

As you say, in ZFC pairs are a defined concept. I can pick an interpretation of the primitive concepts of ZFC (set, element-of) and then I know what a pair is, by unrolling the definition as $\{\{a\}, \{a, b\}\}$ (or whatever—it doesn’t really matter).

Might a reader come away with the impression that in MLTT this option has been foreclosed? Because the “definition” is a syntactic one, namely, that a pair is an expression (a, b) where the sub-expressions a and b have the types A and B, respectively.

I guess I'm stuck thinking of the MLTT definition as a syntactic one because it refers to $a\colon A$, that is, a has type A, which is a syntactic concept. This seemed even more evident when you were defining functions, because you started talking explicitly about "expressions" and "free variables".

Posted by: Kevin Watkins on January 11, 2013 12:09 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I don't quite get what you're driving at. Like any formal system, type theory has a syntactic side and (at least potentially) a semantic side. The syntax (a, b) is a formal part of the syntax of type theory, just as the symbol $\in$ is a formal part of the syntax of set theory. When we construct a model (i.e. a semantics) of a formal system, we have to specify the interpretation of each piece of syntax. For instance, when we build a model of ZFC inside ETCS+R, we specify that sets are interpreted by certain well-founded relations, and that the relation $\in$ is interpreted by the existence of an embedding of a certain sort. When we build a model of type theory in a category $\mathcal{C}$, we specify that if two types A and B are interpreted by objects $[A]$ and $[B]$ and two terms $a\colon A$ and $b\colon B$ (in empty context, for simplicity) are interpreted by morphisms $[a]\colon 1 \to [A]$ and $[b]\colon 1 \to [B]$ respectively, then the type $A \times B$ is interpreted by the categorical product $[A] \times [B]$, and the term $(a, b)\colon A \times B$ (which, like A and B and a and b, is a piece of syntax) is interpreted by the unique morphism $1 \to [A] \times [B]$ whose composites with the two projections are $[a]$ and $[b]$.

It's true that type theory does not give you the "option" of defining ordered pairs in terms of set-membership inside the theory, because there is no global notion of set-membership. On the other hand, if we were to build a model of type theory inside a model of set theory, such as by taking the category of sets in a model of ZFC, then we would have to specify a particular categorical cartesian product of the set-interpretations of two types, and at that point we would probably make some choice like Kuratowski's.

If that doesn't answer your question, then maybe you can ask it again differently. (-:

Posted by: Mike Shulman on January 11, 2013 7:03 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Moving to the northeast just a little, let's be a tad clearer: The Kuratowski pair (for example) $\{\{x, y\}\{y\}\}$ "is" *the* set (if there is one) whose elements are the sets (if there are such) whose elements are &c. That there are always such sets uses the unordered pair axiom, thrice; that this recipe pins down a particular point of V is extensionality, again used thrice. Recovering x and y from their Kuratowski pair is an exercise in moving bits around.

Yes, the Kuratowski pair is definable with parameters x, y; but something else is needed to assert that there *is* such a thing, a construction axiom.

The main difference (I can see) in MLTT is that while you can construct a pair $(x, y)\colon X \times Y$ for $x\colon X$ and $y\colon Y$, the type $X \times Y$ itself *is not definable* without reference to the constructor, (in coq style)

```
Inductive product X Y := | pair : X -> Y -> (product X Y).
```

One of the four kinds of axioms Mike mentions in the write-up then allows you define something like the recovery of x and y from (x, y), which is really what pairs are for; but there are plenty of other ways to define types that will do as product types for you. For instance,

```
Inductive two := | left | right.
Definition twoTypes := two -> Type.
Definition product' (tt : twoTypes) := forall s : two, tt s.
Definition product'' (X Y) := product' (fun s => match s with left => X | right => Y).
```

One can then go on to prove that `product'' X Y` and `product X Y` are (weak homotopy) equivalent; as it happens *assuming univalence, these two types are then identifiable* but *not canonically*, which is why we shouldn't say they are "the same". Since univalence is consistent $*$, even without assuming it the two types are not distinguishable.

Having now got to this point, it's probably worth noting that *the structure of being a product Type* for `X` and `Y` **is** definable, *assuming univalence*, in the sense that within the type of spans,

```
Inductive span (X Y) := make_span : forall (T), (T -> X) -> (T -> Y) -> span X Y
```

the component of `make_span X Y (product X Y) fst snd` **is** contractible, because the parts of a span tell you exactly what underlying map from its underlying type `T` to the standard `product` will work, which can be an equivalence in at-most contracibly-many ways.

$*$: as consistent as MLTT itself, that is.

Posted by: Jesse McKeown on January 11, 2013 10:15 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> *: as consistent as MLTT itself, that is.

I thought the question of the exact proof-theoretic strength of univalence was still open. I know it is conjectured to be a conservative extension, but has that been proven yet?

Posted by: Ulrik Buchholtz on January 11, 2013 11:46 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

To my knowledge, the direct consistency of univalence relative to type theory is still an open question. (It's not conservative, at least not in the sense that I learned to use that word, since it's an axiom and not an extension of language.)

Posted by: Mike Shulman on January 12, 2013 6:43 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Ah, of course it's not a conservative extension in the sense that the same types are inhabitable (potentially that doesn't even make sense because there could be more types available with univalence than without; in dependent type theory, adding axioms might actually extend the language).

But I was thinking it might be conservative relative to some class of types, for instance just $\mathrm{Nat} \to \mathrm{Nat}$, or the types that represent propositions in first-order (or perhaps higher-order) arithmetic.

Posted by: Ulrik Buchholtz on January 12, 2013 7:44 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

It's an open conjecture (originally formulated by Voevodsky) that every term of type Nat that is definable using univalence is provably propositionally equal (using univalence) to a term that is definable without univalence (and hence, by canonicity, propositionally equal to a numeral). Licata and Harper proved this for 1-truncated type theory with one univalent universe of h-sets, by giving a form of univalence that computes and satisfies canonicity. But as far as I know, the case of untruncated type theory with infinitely many univalent universes is still open.

Posted by: Mike Shulman on January 13, 2013 5:17 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

It's a good point that there are other ways of constructing a type with the (up to homotopy) universal property of a cartesian product, rather than the direct inductive way. But what is non-canonical about the identification of your `product` and `product''`?

Posted by: Mike Shulman on January 12, 2013 6:45 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

1) re consistency claims, my apologies; I might well have inflated a simplicial univalent universe (at some size) to imagine a cumulative chain of univalent Type$_n$s, which I see now is something else. Other subtleties no doubt lurk to foil me elsewhere.

2) any auto-equivalence of `X` gives an auto-equivalence of `product X Y`, so if `X` has separable things in it, or … and `Y` is inhabited, then there are nontrivial equivalences of the underlying product types. As functors (if we may use that word) the equivalence of `product` and `product''` certainly *looks* canonical; but again, that's not the consideration I started with, though it does highlight the interest of functors over ad-hoc constructions.

Posted by: Jesse C McKeown on January 12, 2013 7:19 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

1) The simplicial model does contain a cumulative chain of univalent universes. It's just that that model is constructed in ZFC, so it only shows that univalence is as consistent as ZFC, which is stronger than plain type theory.

2) I'm not sure what your point is. Certainly, autoequivalences of objects induce autoequivalences of their products, but that doesn't make the isomorphism between two different products any less unique or canonical.

Posted by: Mike Shulman on January 13, 2013 4:33 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

re 2) I think my point is just that there is this thing which we're all so used to by now that *being a product type* is not as useful as *being a type with a product structure*; and that univalence is nice (as you have already pointed out in more generality) among other things in that *being a type with a product structure* actually does pin down a contractible thing. But I mentioning this mostly for the benefit of Mr Watkins, who it seemed to me was grateful that particluar pairings were definable things in ZFC.

So I tried to line-up this contrast: ZFC has (various) particular pairings, defined by names; ordinary MLTT has named pairs that are otherwise undefinable; MLTT + Univalence has uniquely definable pairngs in uniquely definable product structures.

Posted by: Jesse McKeown on January 13, 2013 5:23 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> being a product type is not as useful as being a type with a product structure

Why not? I would have said the structuralist point of view says at most that "being a type with a product structure" is *just as good as* "being a product type". And in type theory, even this is arguably no longer true. For instance, with "the" product type $A \times B$, the first component of a pair (a, b) is *definitionally* equal to a, whereas for a type simply "equipped with a product structure" all we could assert is that this is a propositional equality.

> ZFC has (various) particular pairings, defined by names; ordinary MLTT has named pairs that are otherwise undefinable; MLTT + Univalence has uniquely definable pairings in uniquely definable product structures.

Sure, that makes sense, although I don't think it has anything to do with autoequivalences. ZFC and ETCS and MLTT have uniquely definable pairings in uniquely-up-to-unique-isomorphism definable product structures, and the sense in which univalent type theory has "uniquely definable pairings in uniquely definable product structures" is not really any different from this, since univalent equality *is* isomorphism. Just as any nonidentity auto-equivalence of A or B induces a nonidentity auto-equivalence of $A \times B$, in univalent type theory any nonidentity auto-equality of A or B induces a nonidentity auto-equality of $A \times B$.

Posted by: Mike Shulman on January 14, 2013 6:14 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The only distinction I'm making when mentioning "has the type of a product" vs. "is a type with a product structure" is the distinction between $\pi_{-1} Z$ and Z for some dependent type Z. The distinction is at least highlighted by the fact that, first, there are several ways to construct product types in MLTT and *no useful way to test them for equality*, even though the Type Theory admits the question as gramatical; and then, even when two product *types* have the same underlying *type*, they may still be different *products*. That really is all I'm driving at. And I'm pretty sure that you have a sense of not getting what I'm driving at because these points are so natural to you already; Kevin, whom I meant initially to address, seemed to have different concerns altogether. That's all.

Posted by: Jesse C. McKeown on January 14, 2013 10:42 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Okay. (-:

Posted by: Mike Shulman on January 15, 2013 7:52 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

You write

"Personally, I think this aspect of structural-set theory matches mathematical practice more closely. When I say "for all x∈ℝ, x 2≥0", I regard it as a property of every real number that its square is nonnegative. I don't regard it as a property of every thing in mathematics that if it happens to be a real number, then its square is nonnegative. Under the material-set theory reading, one particular instance of "for all x∈ℝ, x 2≥0" is the statement "if the monster simple group happens to be a real number, then its square is nonnegative". I wouldn't expect most mathematicians to naturally regard that statement as part of the content of "for all x∈ℝ, x 2≥0".

On the other hand, sometimes in mathematics, it does make sense to regard "A∈X" as a proposition. For instance, if L is the set of complex numbers with real part 12, then a lot of people would really like to prove that "for all z∈ℂ, if ζ(z)=0 and z is not a negative even integer, then z∈L". Note the difference between the two uses of "∈" in this statement. The quantifier "for all z∈ℂ" is naturally treated as atomic, with z being given as a complex number, rather than as an arbitrary thing with the additional hypothesis that it happens to be a complex number. On the other hand, the statement "z∈L" is a proposition, which might be true or false for any particular z, and is susceptible of proof or disproof (as z∈ℂ is not). Thus, in this statement we see ℂ behaving like a structural-set, while L behaves like a material-set."

You are saying that it is silly to talk about if the monster group is a real number, then its square is positive, since it is self-evident that the monster group is a real number, and thus it does not apply.

1. What if you were dealing with a mathematical object, and didn't know what it was? From the point of view of your current state of ignorance, it could be either a real number or the monster group, so then you could validly say that if it's a real number, then it's square is positive. That would be a legitimate statement. Then let's say you discover that the object is, in fact, the monster group. It was the monster group before you knew what it was. It was, unbeknown to you, the monster group when you had previously made the statement about if it's a real number, then it's square is positive. Were you being silly at that time when you made that remark?

In real life, there would almost never be a circumstance where you were dealing with a mathematical object, and didn't know whether it was a real number or the monster group, but there are plenty of times when you might not know if an object was X or Y, where X and Y are similar but crucially different mathematical objects, and you might make a statement along the lines of "If it is X, then such-and-such is true", and then later you find out, it is actually Y. If you know that it is actually Y, then the previous statement talking about if it were X might been silly in hindsight, but it was a valid legitimate statement at the time. Whether it sounds silly depends on your knowledge.

2. So far, I've been talking about a situation where a mathematical object could be either X or Y, the properties of which are known. You could also have a situation where you know the mathematical object is X, but the properties of X are not yet fully known. It could be a recently invented mathematical object, and mathematicians are still in the process of discovering what it's properties actually are. Then, you might also say, "If X is such-and-such, then such-and-such", but that proposition might turn out not be true, and in reality, it was never true, even before any human was aware of that fact. The original statement was at the time, a perfectly reasonable statement, but as we learn more about it, it suddenly sounds silly. Again whether it sounds silly depends on your knowledge.

It seems that a guiding force in you choosing a foundational system is to avoid silly sounding statements. You rejected material set theory because it supposedly allows silly sounding statements. However, whether a statement is silly sounding, depends on the state of knowledge of the observer, and can change over time, or vary from person to person, so it is subjective and it's better to try to be objective.

3. You make a distinction based on whether the set is a subset of some other set. In your second example, you take it as a premise that Z is a member of C, and ask whether it is also a member of L, which is a subset of C, and you say this leads to a different way of thinking about the problem than if you don't first assume that it is a member of a specific set, and then ask if it's a member of a subset. However, this scenario is equally true in both of your examples. In both examples, you are, in fact, saying that X is a member of set Y, and if it is also a member of set Z, which is a subset of Y, then proposition A is true. In the first example, you have

X - specific example, such as the number 3, or the monster group

Y - mathematical object

Z - real number

A - the square is positive

In your second example, you have

X - specific example, such as 3 + 2i

Y - complex number

Z - L, which is the set of complex numbers with real part 12

A - for all $z \in \mathbb{C}$, if $\zeta(z)=0$ and z is not a negative even integer, then $z \in L$

So, you see in both cases, you are starting with a set, and saying "if it is also a member of such-and-such subset, then such-and-such is true". The reason you didn't think that was what you were doing in the first example is because "mathematical object" is such a broad set. Well, who decides how broad a set had to be, before it is so broad it doesn't count as a set? There are circumstances, where the set of all complex numbers, would over broad, such as if the number you are looking for is the number of beers in your refrigerator. Again, this depends on the observer. There are situations where one person might think that it is self-evident that X is in category Y, and that didn't need to be stated, and therefore think of it more like your first example. A another person might think it is not self-evident that X is in category Y, and think that should be stated, and therefore think of it as more like your second example.

It is subjective whether a given mathematical statement is silly or self-evident, or whether a set is a subset of another set. In your post, you selected examples where most people would agree, but I'm sure there are cases in mathematicians, where reasonable mathematicians will not in general agree.

Posted by: Jeffery Winkler on January 7, 2013 10:37 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I think you're misunderstanding the sorts of silly questions which make structuralists uncomfortable. Perhaps a better example is the proposition “$3 \in \pi$”. If you ask an analyst whether this proposition is true, they'll immediately say “that makes no sense”. The answer to such a question has nothing to do with 3 or π as things in themselves, or even as things in $\mathbb{R}$—it's a question about a particular encoding in a particular universe. The truth of this proposition is not invariant under *any* reasonable notion of equivalence weaker than extensional equality.

I'm not sure how best to express the rest of my response, but it has something to do with conflating the notions of “X satisfies these properties” and “X is in some set”:

When I say “Let G be a group, then $\varphi(X)$” I mean something different (philosophically) than when I say “Let $z \in \mathbb{C}$, then $\psi(z)$.” Perhaps someone more eloquent (and more experienced) can pick this up and turn it into an actual response.

Posted by: Cory on January 8, 2013 1:08 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> When I say “Let G be a group, then $\phi(G)$” I mean something different (philosophically) than when I say “Let $z \in \mathbb{C}$, then $\psi(z)$.”

Interesting! I don't. At least, not when I'm thinking like a type theorist. The quantifier “Let G be a group” means “let $G : \mathrm{Grp}$” where Grp is the type of groups, while “Let $z \in \mathbb{C}$” means “let $z : \mathbb{C}$” where $\mathbb{C}$ is the type of complex numbers.

Of course, there are size/universe issues in considering the type of groups (all groups vs “small” groups) which don't arise for the type of complex numbers, but it doesn't sound as though that's what you have in mind. Or is it?

Posted by: Mike Shulman on January 8, 2013 4:11 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

What I understand about Cory's example is that the former case could be a statement about the internal structure of the group, without reference to any other group. The latter case is a statement (presumably) about how this complex number is related to other complex numbers, not about its internal structure (of which it has none).

But perhaps my view is too simplistic. From a categorical viewpoint, I suppose, a group has no internal structure, and the only properties that it has are those that determine how it relates to other groups.

Posted by: Tom Ellis on January 8, 2013 7:30 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I think “At least, not when I'm thinking like a type theorist” is at the heart of it:

Informally, when I ask for a group, I'm asking for a structure with certain properties; I'm not asking for an element of some collection. The fact that we *can* talk about the [your favorite word for collection] of all groups is incidental: I would continue to ask for groups even if we couldn't collect all of them into a single "set".

On the other hand, when I ask for a real number (or an element of a group), I'm asking for something which I know *lives* somewhere particular.

Perhaps another way of expressing this is that I think of *structures* as intensionally defined: I only care about what properties my structure satisfies, but when I think of *elements*, I expect them to live in some "extensionally"* defined collection: I feel I ought to be able to choose one element off of a list.

Sure, when I'm doing type theory, and I happen to want a group, I'll choose something out of the type Grp, but this as an issue of formalization within a particular system; I certainly don't think "Let me go over to the type Grp and grab an element" when I'm doing "everyday" mathematics.

*Extensional is in quotes, because that isn't *quite* correct, but I'm not sure how to express it better.

Posted by: Cory on January 8, 2013 11:35 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Isn't that to interpret the type Grp materially?

I should think that once you see $\mathrm{Grp}: \mathrm{Type}$ defined, and then how you can reason from the type declaration

$$G : \mathrm{Grp},$$

all will be as you desire.

Posted by: David Corfield on January 8, 2013 2:09 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I'd love to see $\mathrm{Grp}: \mathrm{Type}$ defined! I think I've been treating type theory has "just another set theory", but seeing that might clarify where that intuition breaks down.

Posted by: Cory on January 8, 2013 7:15 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

This is a very interesting point. Take type theory without a universe. (I don't really know what I'm talking about here, so humor me and pretend like I'm saying something intelligible.) You can embed intuitionistic first-order logic into the type theory, but you can't quantify over the whole universe. A statement like "there exists a group such that blah, blah" becomes a metatheorem, while a statement about elements is a theorem. Even if you add in universes, you can only internally prove things relative to the universe, right?

Posted by: Walt on January 8, 2013 2:48 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

This is a very interesting discussion, which I think is greatly clarifying in my mind the issues involved. It seems to me that there are two very different things going on here.

Firstly, there is the question of universes. For purposes of this post, I have no interest in type theory without universes. In fact, we might as well assume there is one universe which contains all types, including itself. Technically this is an inconsistent assumption, but there are **techniques** [http://golem.ph.utexas.edu/category/2012/12/universe_polymorphism_and_typi.html] that allow us to usually speak as if it were true.

Secondly, and more importantly, there is the distinction between things that are defined "internally" in terms of how they are built, versus things that are defined "externally" in terms of what properties they satisfy or what collections they belong to. I hesitate a bit to identify this distinction with "material" versus "structural", but there's certainly a close connection: in material set theory, everything is built somehow even if you choose to ignore it, whereas in structural set theory everything is characterized only externally. We *can* always choose to ignore how something was built and consider only some specified collection of properties that we know about it, or some specified collection that it belongs to; I think this is part of Walt's point about doing mathematics structurally inside ZFC.

In type theory, we do have types whose elements are "built" and have internal structure that we can use. In fact, all *particular* types are like that. So when I say $G : \mathrm{Grp}$, that means that G is a data structure consisting of a type (usually denoted also by G, using an implicit coercion), a function $m : G \times G \to G$, an element $e : G$, and a few axioms. That means Grp is defined something like this:

$$\sum_{G:\mathrm{Type}} \sum_{e:G} \sum_{m:G\times G\to G} \mathrm{GroupAxioms}(G, e, m).$$

The situation is, however, the same with complex numbers, if we assume them to have been defined in some concrete way such as "pairs of real numbers": if $z : \mathbb{C}$, then z is a data structure consisting of elements $x : \mathbb{R}$ and $y : \mathbb{R}$. Similarly, if we assume $\mathbb{R}$ to have been defined in some concrete way, then x and y are themselves data structures consisting of some stuff built out of rational numbers.

We may choose to *ignore* that internal structure on elements of any type. And we may choose to *force* ourselves to ignore it by working more abstractly, regarding the type itself as an element of some other type that defines it more or less precisely. For instance, rather than giving a concrete definition of $\mathbb{C}$ and then working with $z : \mathbb{C}$, we might instead assume some element $C : \mathrm{AlgebraicallyClosedFieldOfCharacteristicZero}$ and then work with $z : C$. (Of course, C is itself a data structure consisting of a type, field operations on that type, and axioms; in writing $z : C$ we again use an implicit coercion from such data structures to their underlying type.) In this case, we can't break apart $z : C$ into its real part and its imaginary part; we can only use the structure on C specified by the concrete type it belongs to.

A concrete construction of $\mathbb{C}$ then becomes a proof that the type AlgebraicallyClosedFieldOfCharacteristicZero is inhabited. We can of course choose a different "signature" or "interface" depending on what we need, which may or may not characterize its elements uniquely; sometimes we may need only AlgebraicallyClosedField, other times we may want AlgebraicallyClosedFieldOfCharacteristicZeroAndTheCardinalityOfTheContinuum.

We can do the same thing with Grp just as well as with $\mathbb{C}$. The type Grp underlies data structures belonging to types such as BicompleteCategory, AlgebraicCategory, SemiAbelianCategory, and even some which characterize it uniquely, such as EilenbergMooreObjectInCatForTheFreeGroupMonadOnSet. If instead of defining Grp in a concrete way, we merely assume it to be an element of one of these types, then the assumption $G : \mathrm{Grp}$ doesn't allow us to break G down into a type, a multiplication, a unit, and its axioms; instead we can only use the structure on Grp that we obtain from knowing the type it belongs to.

This sort of process of abstraction is, of course, basic to mathematics. Category theory differs, perhaps, in that it tends to carry out the abstraction at one higher stage: while most mathematicians are comfortable with the idea of characterizing $\mathbb{C}$ abstractly as an algebraically closed field etc., it is less familiar to characterize Grp abstractly in an analogous way. But in category theory, this can be a very useful thing to do.

So I think the difference between groups and complex numbers that Cory is pointing out is not an *intrinsic* difference, but rather a difference in the way that most mathematicians are used to treating them. One of the merits of type theory is that it allows us to operate with "higher level abstractions" of this sort in the same way that we operate with lower level ones.

In fact, this is all really a mathematical counterpart of data abstraction in programming. Perhaps some computer-scientist will correct me, but I think one might say that what we're talking about is the difference between *particular* types, whose elements are built in some specified way, and *abstract* types, whose elements are not.

Material and structural set theory are trying to deal with this issue in different ways. Structural set theory effectively has *only* abstract types. This makes the process of abstraction easy and convenient, but it's then cumbersome when we want to talk about particular types. Material set theory is better in this respect, as Francois and Walt have been pointing out: just as in MLTT, we have "only" particular types (i.e. sets), but we can define particular types whose elements are abstract types, e.g. the ZFC-set $\{(G, e, m) \mid G$ is a set, $e \in G$, and m is a function $G \times G \to G$ satisfying the axioms of a group $\}$. The problem is that when we want to characterize something uniquely but abstractly, in material set theory we can only do it up to isomorphism, so we have to carry around those isomorphisms explicitly all the time and verify that everything we do is invariant under isomorphism. This is one thing structural set theory is trying to avoid, by forbidding operations that are not isomorphism-invariant. But univalence is even better, since it allows an abstract characterization to be truly unique: any two constructions of the complex numbers are *equal*.

Posted by: Mike Shulman on January 8, 2013 7:31 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

(1) Classes and interfaces in Java (and similar constructs in other programming languages) would correspond to the particular and the abstract that you are talking about.

(2) You cannot construct the set of all groups in ZFC, otherwise it would contain a group of every cardinality, and hence the union of all the groups would be larger than every cardinality, which is a contradiction over ZFC.

(3) It is deeply unsatisfying to not have type itself being a type and not have a universal type. I think that neither ZFC nor ETCS nor type theory satisfactorily answers to this.

Posted by: David on June 4, 2015 9:18 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> Classes and interfaces in Java (and similar constructs in other programming languages) would correspond to the particular and the abstract that you are talking about.

Hmm, I think I would have said that classes and interfaces are just particular types built using the universe. By an "abstract type" I mean the sort of type that occurs in a parametrically polymorphic function like `concat : forall a, List a -> List a -> List a`.

> You cannot construct the set of all groups in ZFC

I mentioned at the top of this comment that "we might as well assume there is one universe which contains all types, including itself. Technically this is an inconsistent assumption, but there are techniques that allow us to usually speak as if it were true", so that the spirit in which I meant "the set of all groups" to be read.

> It is deeply unsatisfying to not have type itself being a type and not have a universal type. I think that neither ZFC nor ETCS nor type theory satisfactorily answers to this.

Well, "deeply unsatisfying" is a value judgment that different people might make differently. Personally, I would probably say that I find it deeply unsatisfying only every other Thursday. It's true that neither ZFC nor ETCS has a "solution" to this problem. There was a while when I thought that type theory could sort of answer it, but I'm not so sure any more.

Posted by: Mike Shulman on June 5, 2015 5:48 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

This came up in the other thread, but I don't think that $3 \in \pi$ or its negation is a real example. The correct, even structural, definition in ZFC is that there exists a set with certain operations which has an element 3 and an element π with whatever properties determine them. There exists a set where $3 \in \pi$, and a set where $3 \notin \pi$, so ZFC doesn't assigns a definite truth value to it the way it does $1 + 1 = 2$. A better example is $\exists x. x \in 3 \vee x \in \pi$.

Posted by: Walt on January 8, 2013 1:53 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I don't see what makes that question a better example (although it certainly isn't worse!). Your same argument can be made about that question: there are instances of $\mathbb{R}$ where $\exists x. x \in 3 \vee x \in \pi$ and there exists instances where the negation holds.

But that's actually the point: $\mathbb{R}$ *ought* to be defined structurally, precisely because both my question and yours don't make sense, but as far as I can tell, there's no good way in ZFC (or any material set theory?) to define $\mathbb{R}$ structurally: there are two obvious constructions of $\mathbb{R}$, as a set of equivalence classes of Cauchy Sequences of rationals, or as the set of Dedekind cuts of rational numbers. Both give definite truth values to "silly" questions, but these answers are *not* properties of 3, π or $\mathbb{R}$, they're properties of the encoding, and passing to a different encoding changes these properties, even though the essential properties of $\mathbb{R}$ are preserved.

The structuralists' argument is that we shouldn't be able to *ask* questions which aren't structurally invariant.

Posted by: Cory on January 8, 2013 6:49 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Walt's point is that in ZFC there is no construction of $\mathbb{R}$ for which $\exists x. x \in 3 \vee x \in \pi$ fails. Since 3 and π must be different, they can't both be the empty set, so at least one of them must have an element.

Posted by: Mike Shulman on January 8, 2013 7:33 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I have no idea what I thought that formula meant when I responded, but I clearly misread it terribly.

Posted by: Cory on January 8, 2013 8:59 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The thing that only just occured to me again is the sort of dialog I often go through to help calculus students write something like arguments; and that is to start by writing definitions, and then use them.

So, if such a *student* was to ask me, challenging this notion they may have heard that "everything is a set", whether $3 \in \pi$ or not, I would start by asking, "what do you mean by 3, π, and $\in$?". I would perhaps also mention that "$\in$" is not a symbol in the language of fields, though it *is* a symbol in real analysis, but that *in* real analysis we *are* careful only to write $\in H$ for H a *set* in $V[\mathbb{R}]$, a universe over a complete ordered field of atoms. Btw, I think this gets at something Mike wanted before: if you want all the real numbers to be atomic in a natural way, expand the language of set theory.

But even if one isn't so awfully careful, such formulae as $\exists x, x \in 3 \vee x \in \pi$ are still about *what do you mean by* $\exists x$, 3, π *and* $\in$ *?* For a student of foundations it may well be very reassuring that they can

- provide a set relatively large enough to support a complete ordered field structure

- reason both internally (in $(0, 1, +, \times, \cdots, \inf)$) and externally (in $\in_V$)

And, to close this note, back to “what do you mean by $\exists x$”: yes, ZFC will assert “$\exists x, x \in 3 \vee x \in \pi$”, but not “$\exists x \in \mathbb{R}, x \in 3 \vee x \in \pi$”. So there.

Posted by: Jesse McKeown on January 10, 2013 6:47 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I think we basically agree on everything, Jesse. When the revolution comes, and they sentence to die by firing squad, our final words will be “What do you mean by ‘sentence’, ‘die’, and ‘firing squad’?”

Posted by: Walt on January 12, 2013 9:39 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> 1 … In real life, there would almost never be a circumstance where you were dealing with a mathematical object, and didn’t know whether it was a real number or the monster group, but there are plenty of times when you might not know if an object was X or Y, where X and Y are similar but crucially different mathematical objects

It would be easier to respond to this (like the rest of your comment) if you gave concrete examples. In all situations of this sort that I can think of, there is an ambient structural-set Z of which X and Y are both subsets.

> 2 … Then, you might also say, “If X is such-and-such, then such-and-such”, but that proposition might turn out not be true, and in reality, it was never true, even before any human was aware of that fact.

Isn’t that the case for *all* false statements, at least if you subscribe to a marginally realist philosophy of mathematics? Not every false statement is silly or meaningless. I don’t see how known or unknown properties of X are relevant.

> However, whether a statement is silly sounding, depends on the state of knowledge of the observer, and can change over time, or vary from person to person, so it is subjective and it’s better to try to be objective.

No, I disagree, at least if by “silly-sounding” you mean “ill-typed statement”, which is the sort of thing that type theory rules out. Whether or not a statement is well-typed ought always to be evident (or, at least, algorithmically decidable) from the definitions of the terms involved. Not every false statement is ill-typed; in fact, most of them aren’t.

> 3 … However, this scenario is equally true in both of your examples.

Only if you insist beforehand on the material-set-theory interpretation of the quantifier. Which I’m saying you don’t have to do. It’s a different way of thinking, I grant you, but my experience is that once you get used to it, you start to see that it’s a more accurate way of describing mathematics.

Posted by: Mike Shulman on January 8, 2013 4:13 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I’ve always been puzzled by univalence. Can someone help me out?

One of the great advantages category theory has given me is to understand the difference between isomorphism and equality. Now univalent type theory is making them one concept again!

Hopefully the solution is referred to by the phrase ‘“equality” in univalent type theory behaves a bit differently from what you are probably used to’. Can you explain more about this?

Posted by: Tom Ellis on January 8, 2013 7:23 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Would it make you feel any better if I said instead that univalent type theory is throwing out the notion of "equality" (at least between objects of categories) — because almost always what you want in practice is isomorphism instead — and then re-using the word "equality" to mean "isomorphism"?

Posted by: Mike Shulman on January 8, 2013 7:38 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Yes Mike, that is reassuring! I suppose in type theory "equality" means something very different to what I am used to.

Posted by: Tom Ellis on January 8, 2013 8:33 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Thanks for a very nice post, Mike. There were various things I was going to say in response. One was what Kevin Watkins said about functions being "expressions", but I think you've cleared that up. Another was what Tom said here. In your post, you said that the univalence axiom

> can be concisely described as saying that "isomorphic structures are equal".

If I hadn't been given the context, I'd have called that not univalence but skeletality, which is generally seen as a distraction at best and "evil" at worst. But I gather something quite different is meant, and your reply to Tom's comment confirms this.

I find myself wanting to ask the same questions as **Karol** [http://golem.ph.utexas.edu/category/2013/01/from_set_theory_to_type_theory.html#c043053] . As far as broad exposition is concerned, redefining the word "equality" seems a terribly risky move. Asking people to change their foundational outlook is already enough of a big deal, without playing with the meaning of this extremely primitive concept!

Or is it the case that we won't really understand what's going on unless the word "equality" is used? I mean, if for the purposes of explaining to beginners you said "equivalent" (or something) instead of "equal", would that somehow give a false impression?

Posted by: Tom Leinster on January 8, 2013 7:23 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I wonder if structuralism actually commits you to univalence? The whole point of structuralism, if I understand it, is that anything that you're not supposed to say you make literally unsayable. But the idea of "evil" violates that – you can ask perfectly well "Are there three one-element sets in the category of sets?", even though you're not supposed to. You can't make this unsayable, since you have to be able to write formulas for object equality to express that two arrows are composable, so the next best thing would be to say there's only one.

Posted by: Walt on January 8, 2013 10:14 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> You can't make this unsayable, since you have to be able to write formulas for object equality to express that two arrows are composable

Actually, you can do this with **dependent types** [http://golem.ph.utexas.edu/category/2010/03/in_praise_of_dependent_types.html] . If the type of arrows is dependent on the type of objects, then the composition operation has type

$$\prod_{x,y,z:\mathrm{Obj}} \mathrm{Arr}(y, z) \times \mathrm{Arr}(x, y) \to \mathrm{Arr}(x, z)$$

which doesn't need any equality of objects.

Posted by: Mike Shulman on January 8, 2013 10:19 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

(Something needs turning upside-down there.)

Posted by: Tom Leinster on January 8, 2013 11:27 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

No, I think it's right, but maybe it would be clearer with some parentheses:

$$\prod_{x,y,z:\mathrm{Obj}} \left(\mathrm{Arr}(y, z) \times \mathrm{Arr}(x, y) \to \mathrm{Arr}(x, z)\right)$$

Posted by: Mike Shulman on January 9, 2013 5:51 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

About your last paragraph, why do you use the word "equal" to refer to a concept much more complex than what an average mathematician understands by equality? There are already words whose established meaning is much closer to the intended meaning like "connected by a path" or "homotopic". When you say "A is equal to B" it sounds like something which is either true or false and carries no more information i.e. should be an h-proposition, while in fact it can be an arbitrarily complicated type.

Just from the comments to this post I see that this linguistic issue creates some confusion. What's the justification for this choice of words?

Posted by: Karol Szumiło on January 8, 2013 7:59 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

To Karol, Tom, and anyone else feeling the same way: believe me, I've been where you are!! For a long time I resisted calling this thing "equality", for exactly the reasons you give, and more. And I agree that when you're starting out, it can be better to think of it as "homotopic" or "isomorphic" or "equivalent". I don't think those words are *wrong* in any way. But over time I've been converted to believe that in the long run, once people are familiar with univalent foundations and doing mathematics in it, we have to *also* be willing to call it "equality".

First of all, in set-theoretic foundations, we need to distinguish between equality and weaker forms of sameness. But in type theory, the "identity types" are the *only* notion of sameness we have. (Well, there is "definitional equality", but it doesn't serve all the same purposes, being a judgment rather than a proposition.) There is nothing stricter than homotopic/isomorphic/equivalent for which we need to reserve the word "equality". So at least *after* you're familiar with the system, there is no potential for confusion in using the word "equality".

In particular, that means that sometimes — arguably even *most* of the time — we do want to call the identity type "equality". For instance, when I say that for all $x, y: \mathbb{R}$ we have $x + y = y + x$, that equals sign is the identity type; in type theory this statement would be the type $\prod_{x,y:\mathbb{R}} \mathrm{Id}_{\mathbb{R}}(x + y, y + x)$. Given the prevalence of equalities in mathematics, this means that very frequently, we have to pronounce the identity type as "equals" if we want our mathematics to look vaguely like traditional mathematics.

It's true that in most or all of these cases, the identity type being pronounced as "equals" is one that *does* behave more or less like traditional equality. In particular, these identity types contain at most one element, so there is no "extra information" contained in a proof of such an equality. In HoTT, we say that these identity types are "h-props" or "mere propositions", and that the underlying type is an "h-set". So you might reasonably argue that we should only pronounce the identity type as "equals" in this special case.

There are a lot of reasons why I've come to believe otherwise, and I'm not sure I'll be able to think of them all right now, but I'll give it my best shot and maybe more will come to me as we talk (and others can hopefully also chime in). This is of course a really important issue, and we'll need to think carefully about how to exposit it in the introductory stuff we're writing, so it's good that we have this conversation now!

Firstly, I feel that reserving "equals" for the case of h-sets may give the wrong impression that there are two separate concepts — equality and homotopy/equivalence/isomorphism — when in fact there is really only *one* concept. Coming from a set-theoretic background, one naturally expects that in addition to a notion of "isomorphism" between structures, there is also a notion of "equality", but this is not the case in univalent foundations. In particular, we can't write down strict versions of higher-categorical structures. Separating "isomorphism" from "equality" linguistically seems to me to be liable to promote this confusion.

Secondly, you might be surprised at some of the things that even this restriction would allow. For instance, consider the type of natural number objects discussed above:

$$\sum_{N:\mathrm{Type}} \sum_{z:N} \sum_{s:N\to N} \mathrm{PeanoAxioms}(N, z, s).$$

This type is contractible, and therefore, in particular, it is an h-set. Thus, even if you restrict "equals" to the h-set case, it will still be true that any two natural numbers objects are not just homotopic but *equal*. Another nice example is the type of well-ordered sets: since an isomorphism of well-orderings is unique when it exists, this type is an h-set, and so any two isomorphic well-orderings wil be not just homotopic but equal. On the other hand, if we relax any of the conditions on these types, then they are no longer h-sets; but it seems strange to pop back and forth between using words like "homotopic" and "equal" for what is really the same concept (the identity type).

Thirdly, once I got used to the idea, I found it tremendously freeing. As a category theorist, it's a constant temptation to me to write $=$ rather than $\cong$ or $\simeq$ when I'm calculating some long string of isomorphisms or equivalences, and I know some people do do that, at least in informal settings.

Fourthly, if you haven't yet, read **this ongoing discussion** [http://golem.ph.utexas.edu/category/2012/12/what_can_category_theory_do_fo.html#c042928] . I find it very pleasing how univalence can make precise philosophically "natural" ideas that look totally bizarre from a set-theoretic perspective. Univalence really is a different way of thinking about mathematics, and feeling uncomfortable at first is good, because it means you're less likely to carry over wrong expectations.

Fifthly, if we allow ourselves to call all identity types "equality", then there are intriguing and useful points of contact between homotopy type theory and the traditional "propositions-as-types" interpretation of type theory. For instance, in homotopy type theory, the proposition (= type) which asserts that a type A is contractible is

$$\mathrm{isContr}(A) := \sum_{a:A} \prod_{b:A} \mathrm{Id}_A(a, b).$$

Read homotopically, an inhabitant of this type consists of a point $a : A$ (the center of contraction) and a continuous deformation from the identity function of A down to the constant function at $a : A$. But read in the traditional propositions-as-types style, this type would be the assertion "there exists an $a : A$ such that for all $b : A$ we have $a = b$", i.e. the assertion that A is a *singleton*. Other definitions and proofs in HoTT can also be interpreted in this way, which gives interesting insight into both the homotopy theory and the traditional type-theoretic perspective — but only if we're willing to also call the identity types "equality".

And lastly, even when the identity type is an h-prop, it doesn't behave exactly like equality in set theory. Equality in type theory is *not* a "primitive concept" in the same way that it is in set theory; you *have* to learn to think about it differently and treat it differently, even prior to the introduction of univalence. In other words, we are already "playing with the meaning" of equality by passing from set theory to type theory, and there's no way around that. So telling people that the word "equality" means something different may be a good thing, even if it is scary at first, because it will force them to come to grips with the fact that even in the more familiar-looking cases, equality behaves differently than they may be used to.

Posted by: Mike Shulman on January 8, 2013 10:12 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Thanks for a detailed answer. I want to say that on the level of mathematics I find your arguments convincing (with some exceptions discussed below). But what I really had in mind are practical considerations. If HoTT were to be universally accepted as a foundation of mathematics tomorrow, I wouldn't resist your nomenclature. However, the best you can count for is that it will coexists with other approaches. Do you really want your notion of equality to compete with the traditional one? I am afraid that such a competition would only alienate many potential users of HoTT.

This rather pessimistic point of view comes from an observation that the differences between the two notions seem to be even subtler than what we discussed so far. Overcoming those subtleties requires a lot of motivation, effort and getting used to new ways of thinking. Looking at my own example I see that whenever I read something about HoTT I need to mentally substitute "path" for "equality" and "path object" for "identity type" and only then I see what's going on. So for example when I read something like "every two natural numbers objects are equal" I need to explicitly remind myself that this also means that every two isomorphisms between natural numbers objects are equal and that every two "homotopies" between such are also equal etc. On the other hand when I read "the type of natural numbers objects is contractible" the complete picture immediately pops into my mind and I don't need to spend any extra time to understand this statement. I think the problem here is that if you wrote this statement formally in HoTT you would be forced to write down the definition of the contraction of the type of natural numbers objects, but if you say it informally in English this specification is lost somewhere between the lines.

Another example of this phenomenon has its origin in grammar. The word "equal" is an adjective which is used in natural language as if it where a mere proposition. So saying "A is equal to B" to refer to type theoretic equality leaves a lot of implicit information to be derived when reading such a statement. In traditional mathematics if I want to be precise about what I say, I will simply refrain from using phrases like "A and B are isomorphic". Instead, I will explicitly define a morphism f and then state that "f is an isomorphism from A to B". Again we are in the situation when writing this formally in HoTT would force us to write such an isomorphism explicitly. On the other hand in informal HoTT an innocent statement like "A is equal to B" has a lot of baggage attached to it and we are not given any hint that we need to remember about this baggage. It looks like formal HoTT handles this situation explicitly and by design while informal HoTT makes it even worse. That's unless there is some way of talking about HoTT informally in English which handles all this baggage properly.

Let me now address some specific arguments in favor of your terminology.

> I feel that reserving "equals" for the case of h-sets may give the wrong impression that there are two separate concepts — equality and homotopy/equivalence/isomorphism — when in fact there is really only one concept.

It happens in mathematics that (important and frequently used) special cases of notions are given special names and it doesn't seem to create much confusion. For example we use "vector space" to refer to a module in the special case when the ground ring happens to be a field. Surely, "vector space" and "module" are not separate concepts but situations when we work over a field often have some special flavor which is emphasized by this choice of words. I've never observed this giving anyone an impression that vector spaces and modules are two separate concepts. It seems to me that the situation with "traditional equality" and "homotopy type theoretic equality" is not dissimilar.

> Other definitions and proofs in HoTT can also be interpreted in this way, which gives interesting insight into both the homotopy theory and the traditional type-theoretic perspective — but only if we're willing to also call the identity types "equality".

I actually followed a similar line of thought in order to argue something opposite. If we can interpret some language in two different ways, it's not unnatural to name some concept differently in those two interpretations. For example the binary operation of the language of groups is called "multiplication" in some interpretations and "addition" in others. While there is no formal distinction between those interpretations, this choice of words definitely highlights differences in the intended way of using the notion of group. Similarly, since "homotopy type theoretic equality" is much subtler than the classical one I would say it deserves to be given a separate name to highlight this subtlety.

Of course there is also a philosophical question of whether the homotopy type theoretic notion of equality is somehow better or addresses mathematicians' needs more accurately than the traditional one. Dedekind's conception of equality mentioned in another topic you linked suggests that something like this may be the case. But even if this is true, do you really want to attempt both foundational and philosophical revolutions at the same time? I feel that trying to overthrow the traditional concept of equality would only hinder the efforts to promote type theoretic foundations. On the other hand I have to admit that my resistance to your nomenclature may partially come from the fact that I'm at the point where I don't think about HoTT as foundational system for mathematics but merely as a language of homotopy theory. Thus for me an identity type is a homotopy theoretic concept and not a foundational one. Perhaps if were thinking about those things in more foundational terms I would be more likely to accept your point of view.

Posted by: Karol Szumiło on January 9, 2013 7:33 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

For whatever it's worth, since they changed all the notations in the coq HoTT from ~~> to =, I've been reading `p : a = b` as "let `p` be an identification of `a` and `b`".

Posted by: Jesse C. McKeown on January 10, 2013 4:50 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Thanks for the thoughtful reply! I'm definitely open to being convinced back to your side, but I'm not convinced yet. I think you're right, though, that the two arguments you replied to specifically could cut both ways.

> Do you really want your notion of equality to compete with the traditional one?

I think what I'm saying is that we don't get a choice about this. The notion of equality in type theory *is* different from the traditional one, no matter how we slice it. So we should bite the bullet and call a spade a spade (and why not mix a few metaphors while we're at it). And it's not just equality: logic is different too.

For me, I think one of the biggest hurdles to get over was learning to think in what Andrej calls *proof-relevant mathematics*. In type theory, there is really no such thing as something being "just true": *every* time you assert some proposition you must give a proof witness, and every time you use an assumption you are handling such a witness. And so every proof you write is a program that can be executed, transforming input witnesses to output witnesses. This is where a big part of the power of type theory comes from.

Once I really internalized that, then it started to seem perfectly natural that there could be multiple different proofs — different witnesses — to an equality. Univalence just says that in some important cases, those witnesses are isomorphisms. The point is that it's not really *equality* that's behaving nontraditionally — the whole system of mathematics has become "proof-relevant", and equality is just a particular manifestation of that. In other words, *everything you say* has what you call "baggage" attached to it; so we don't need a hint to remind us of that fact, because it's *always* true.

Of course, it may happen, sometimes, that every witness to a given equality is provably equal to any other such witness (i.e. the underlying type is an h-set), but I think there's no particular reason to privilege this situation with a special word. More importantly, even in this case, equality is *still* proof-relevant: we may care about the particular witness, even if every other witness is equal to it. For instance, the assertion that a given function f is an equivalence is an h-proposition, but a witness to that fact includes the data of an inverse to f, and we often care what the inverse of a function is.

> I don't think about HoTT as a foundational system for mathematics but merely as a language of homotopy theory

This may be another hurdle that is important to get over, especially since in classical homotopy theory, there *is* another, stricter, notion of "equality".

Posted by: Mike Shulman on January 10, 2013 5:54 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I believe I'm slowly starting to see where you're coming from with your motivation for this particular choice of terminology. But I'm still afraid that it will resonate rather badly with most mathematicians.

> In type theory, there is really no such thing as something being "just true": every time you assert some proposition you must give a proof witness, and every time you use an assumption you are handling such a witness.

I am actually very sympathetic to this idea because I see it as a formalization of a certain "good practice" in mathematics – the one I referred to in my "A is isomorphic to B" example in my previous post. I surely don't fully understand what it entails when it is taken to such an extreme that even equality is made proof-relevant.

> In other words, _everything you say_ has what you call "baggage" attached to it; so we don't need a hint to remind us of that fact, because it's always true.

Right, but there is not only a matter of remembering that "baggage" is present but also specifying what this "baggage" actually is. This issue is deeper than just choice of one word over another. It seems that English grammar is simply insufficient for proof-relevant mathematics. We would need to invent a new part of speech that is to an adjective like (roughly speaking) a general type is to a proposition and whose usage automatically entails specifying a witness to whatever it describes.

> This may be another hurdle that is important to get over, especially since in classical homotopy theory, there is another, stricter, notion of "equality".

That's true but what I find attractive about HoTT as a language of homotopy theory is that it should allow me to talk solely in homotopy invariant ways. If I really knew how to do it I would gladly wave the classical strict notion of equality goodbye. For example I would like to have a ready to use formalization of general homotopy colimits in HoTT. At the moment I only know how to deal with homotopy colimits in specific models which heavily rely on traditional notion of equality. I would even say that this is the main reason which makes me cling to strict equality.

Posted by: Karol Szumiło on January 11, 2013 1:11 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I am more optimistic. It will take a while, certainly, since we aren't yet at the point where univalent foundations can offer many concrete advantages to mathematicians. But I believe we will get to that point sometime in this century, and then pragmatism will gradually take over. Probably there will be a large number of people who will never accept univalent foundations, but eventually the younger generations will grow up with the ideas and the revolution will be complete. (-:

I am intrigued by your idea of inventing a new part of speech! I haven't really found English grammar inadequate for proof-relevant mathematics so far, though; generally I use "propositional language" (e.g. "a equals b") when I don't care about naming the witness (even though it is always implicitly present), and when I do want to name it, I say something like "p is an equality between a and b". Jesse's suggestion "p is an identification of a and b" is nice too, and could perhaps be shortened to "p identifies a and b". More generally, one can say things like "p is a proof of P" or "p proves P".

Posted by: Mike Shulman on January 11, 2013 7:11 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

This is probably just me, but I find this kind of analysis unattractively messianic. It really does exactly resemble the confident predictions of revolutionaries 100 years ago about New Socialist Man. Something about foundations makes people jump immediately from personal enthusiasm to a desire to convert the entire world to a new world view. On some level you must recognize what it sounds like, since you've joked about revolution a couple of times already.

Posted by: Walt on January 11, 2013 8:18 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Sorry; I guess my smiley face wasn't sufficiently clear that the talk about revolution was entirely a joke. I'm actually more of a **pluralist** [http://math.andrej.com/2012/10/03/am-i-a-constructive-mathematician/] . If there is to be any revolution, it should probably be an overthrowing of the view that any *one* foundation ought to suffice for all of mathematics.

Posted by: Mike Shulman on January 11, 2013 9:32 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Was the whole first paragraph a joke? I took the smiley face to be applying only to the final "and the revolution will be complete". We need a smiley face scoping operator to remove this kind of ambiguity.

I would have called myself a pluralist, but after I read Andrej' blog post, now I'm not so sure, mainly because he's puts the emphasis on places I find odd. His n different notions of the reals are n different concepts, so no conflict between them is possible, and any conflict is an illusion. To phrase it as "different mathematical worlds" is to put the conflict center stage. It's like we decided to call every ring "the integers" and said things like "imagine a world where there was an integer x such that 2*x = 1".

Posted by: Walt on January 11, 2013 10:36 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

No, the rest of the paragraph was mostly serious. But that shouldn't scare you; it's just a description of how all new abstract ideas percolate into mathematics. At first, the new idea doesn't offer very much except maybe a new way of looking at old things, but gradually people start to see that it has real benefits.

Regarding pluralism, it doesn't sound to me like you quite understood what Andrej was saying about the reals. There is *one* definition of the reals which behaves very differently depending on the foundational system you are in. (Actually, of course, there is more than one definition of the reals, but each of them also has this behavior.) Does the definition of an "equivalence class of Cauchy sequences of rationals" become two different *concepts* because it behaves differently depending on whether we assume the law of excluded middle or the axiom of countable choice?

Thinking of foundational systems as analogous to rings is a very good analogy. But in that case, "the reals" would have to be replaced by something definable relative to any ring, like ideals. Ideals behave very differently in different rings depending on the properties of the ring, but the concept of "ideal" is still the same.

Posted by: Mike Shulman on January 12, 2013 6:27 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

There's a banal version of your analysis, and there's a messianic version. The banal version would be "I think these techniques are exciting, and I think they'll prove useful in the future." The messianic version is Kruschev pounding his shoe on the table at the UN and shouting “we will bury you!” The first one is attractive, because I like exciting ideas that might prove useful in the future. The second one is unattractive telling me that my children and my children’s children will be homotopy type theorists whether I like them or not, so I should make my peace with it. I try to restrict my “making peace” activities to death and politics, and keep it out of mathematics.

I’m not clear if Andrej meant what you said or not. I suppose this could be another semantic disagreement, but if there’s a unitary concept with different examples, then I would say “a”, rather than “the”. “A ring” has certain properties. There’s no such thing as “the ring”.

To maximize the heighten the semantic confusion, I would call myself a pluralist, and yet I think there should be roughly one foundation of mathematics. That foundation is the list of formal systems we agree are consistent. If we agree on the list of consistent systems, we agree on foundations. We may not agree on notation, but hey, it’s notation. By “ring”, I mean “noncommutative but associative ring with unity”. If you start a paper with “assume rings are commutative, or assume rings don’t have 1,” but it’s notation.

Most arguments about foundations prove, upon careful analysis, not arguments at all. This has been the great unsung discovery of logic and metamathematics in the 20th century. The Brouwerian reals and reals with nilpotents all sit comfortably inside a big shared universe.

Posted by: Walt on January 12, 2013 9:08 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> There’s a banal version of your analysis, and there’s a messianic version.

Well, if I were you, I’d go with the banal interpretation. Going with the “messianic” interpretation, and invoking images of Khrushchev banging his shoe at the UN and firing squads, seems pretty silly to me, and close to manifesting **Godwin’s Law** [http://en.wikipedia.org/wiki/Godwin%27s_law] .

(As a side note, see also **this** [http://en.wikipedia.org/wiki/Shoe-banging_incident] .)

Time will tell whether HoTT is a flash in the pan or is recognized as scoring notable successes. I really don’t see any cause for alarm.

Posted by: Todd Trimble on January 12, 2013 3:52 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

It’s not alarm. I have no opinion on homotopy type theory or univalent foundations. I don’t know much about them, and if I knew more, I might think is was totally awesome. I’m reading the thread to see if I can find out about them without going to the trouble of doing actual work. I’m actually a little bit persuaded on univalence by some of Mike’s earlier arguments, though the issue that was raised here about “what do we mean by equality” seems a little bit sticky.

The first person to ever invoke Godwin’s Law on the Internet was Adolf Hitler. True fact.

Posted by: Walt on January 12, 2013 6:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

<smiley scope> I suppose you can always invent something to worry about by misinterpreting what someone else says. And you can always heighten the confusion by using words differently than everyone else. </smiley scope> (-: >

Posted by: Mike Shulman on January 13, 2013 4:32 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

If that's what I'm doing, then I apologize.

Posted by: Walt on January 13, 2013 6:50 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Actually, I want to apologize in general, Mike. I'm probably falling into the trap of reading your comments in the context of the sum total of foundational arguments that I've read over the years. Since you aren't actually responsible for the sum total of foundational arguments – unless you have a really astounding number of pseudonyms – that's not fair to you.

Posted by: Walt on January 13, 2013 8:49 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Thank you! But I'm probably not blameless either. There's something about this subject which seems to automatically make it a flashpoint.

Posted by: Mike Shulman on January 13, 2013 5:32 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

This is true. We should take a solemn vow to only discuss foundations in Loglan or something like that.

Posted by: Walt on January 13, 2013 5:51 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Dear Walt, the way I understand **relativism** (it is not pluarlism, that's what Paul Taylor claimed, but I want to make a distinction below) is more like what Mike describes.

Regarding "the reals" vs. "a reals", in any given world of mathematics there is, up to isomorphism, a single object of reals, so they deserve to be called "the reals". (If you do not subscribe to this opinion, then you will never ever be able to use "the", unless you admit only one mathematical world, but then you're banging with the shoe already.)

There are actually two aspects of realism (and pluralism) in mathematics, both are relevant. The first one is purely mathematical, or shall I say meta-mathematical, where we see that there are different ways of setting up a mathematical foundation (classical vs. intuitionistic, proof-relevant vs. proof-irrelevant, set-theoretic vs. type-theoretic, etc), and we see that all serve useful purposes.

The other one is sociological: people have different ideas on what mathematics is and how its foundation should be set up, analyzed, and philosophised about. These are also useful! Would Gödel have done the things he did if he were a Brouwerian intuitionist? Would Martin-Löf have formulated his type theory if he were a classical Platonist? Would people invent proof assistants if they took Brouwer's position that mathematics must not be formalized?

The two kinds of relativism are entangled in various ways. Keep in mind then that there are two extreme points: the purely logical/mathematical idea that there are many foundations, and the sociological/historical idea that the mathematical activity of humans is not as uniform as one might think.

For me, the magic is in learning how to be able to switch in my mind from one to another kind of mathematics. And by this I mean not only technical proficiency but also the mathematical intuition that allows a mathematician to have a feel for things.

On a less personal level, mathematicians who are not stuck in a single world of mathematics benefit from the plurality of worlds in a number of ways. It is like comparing the experiences and opinions of a man who has traveled around the world in comparison to someone who spent his entire life in a town in the Midwest.

Let me also explain how I take “pluralism” and “relativism” to be two different things. First observe that even if you subscribe to an absolutist view, say you are a hard-core formalist and type-theorist, you will still be able to account for “those other worlds” by building models inside your world. You are a pluralist if you stop here and just say “oh yeah, there are many different ways of doing math, and we cannot really say that one is the true one”.

But there is a further thought you might have: perhaps we can have a grand unifying theory of the different worlds of mathematics in which all these different worlds are analyzed in terms of how they relate to each other etc. The denial of such a possibility is relativism: any point of view of mathematics as a whole is necessarily dependent on the world of mathematics that it is expressed in. You cannot “climb out” or “above” all the worlds to get an absolute view of them all. By virtue of using the mathematical method you will make commitments that will filter your view of mathematical worlds through the view of a particular one. It is like physics: because an observer is somewhere in spacetime, that makes his view of other observers relative.

One possible reaction might be defatist: “Andrej Bauer is saying that mathematics will never achieve harmony. He says it is all necessarily a mess, a matter of opinion.” Another is: “Hey, we need a theory of all this. Wait, he says there is no such theory. Hmm, that is a nice challenge for logicians!”

Posted by: Andrej Bauer on January 13, 2013 8:30 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Interesting. I’m not sure if I subscribe to pluralism. It would depend on understanding concepts like “mathematical world” a little bit better. I’m a little fuzzy on how that is a necessary concept.

I guess if I were forced to say, I’d say a mathematical concept is a set of symbols, a logical language, and the set of theorems that can be produced in the that logical language. From that point of view “the reals in ZFC” is a unique absolute object, as are any of the other “reals” on your list. We can compare them by “interpretability” or any finer relation, like “interpretable preserving the algebraic structure,” or whatever. So it’s natural to describe a set theory or type theory as a mathematical world, but it’s just a big mathematical object that’s good at producing little mathematical objects, like different notions of “reals”. You seem to mean something stronger by it.

Posted by: Walt on January 13, 2013 4:54 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> I’d say a mathematical concept is a set of symbols, a logical language, and the set of theorems that can be produced in the that logical language.

You realize that you just said the word “set”, which of course is a notion that is very dependent on the foundation for mathematics that you choose? (-: If your metatheory includes an axiom like “ZFC is inconsistent” (which of course, by Gödel is as consistent as ZFC is), then the set of theorems you can prove from ZFC is going to be very different than otherwise.

Posted by: Mike Shulman on January 13, 2013 5:30 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

> You realize that you just said the word “set”, which of course is a notion that is very dependent on the foundation for mathematics that you choose? (-:

Isn't there a famous argument by W. Quine that this in fact applies to any system of higher order logic? In fact, he went as far as to argue that such logical systems, which lack such properties as soundness and completeness do not qualify as *logic* at all.

If there's any argument at all for preferring first-order foundations for mathematics and being skeptical about higher-order systems, I think this may be it.

Posted by: John Smith on January 13, 2013 8:04 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I automatically translate everything into set language. I don't like to boast, but you should see my translation of Keats' *On First Looking into Chapman's Homer* into Venn diagrams. It's exquisite.

I meant "finite set of symbols", and for "set of formulas", I meant the recursively enumerable list of theorems a computer program would generate from the logical rules.

Posted by: Walt on January 13, 2013 8:55 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Dear Walt, translating "everything into set language" (I am assuming you mean something like ZFC) deprives you of the beauty of the world in which the reals embed into the natural numbers, of the world where we can cover the reals with intervals of total length not exceeding 0.42, of the world where every set has at most one complete separable metric on it, of the world in which all real functions are differentiable, and so on. How can a habit be worth more than the immense beauty and richness of those other worlds?

Posted by: Andrej Bauer on January 13, 2013 10:24 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Of course, but depending on whether ZFC is consistent, the statement $0 = 1$ might or might not appear in the r.e. list of 'theorems of ZFC' generated by a computer.

Posted by: Mike Shulman on January 14, 2013 4:19 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Does that matter? Nothing in my proposal requires a stand on whether ZFC is consistent. I mean "X is a theorem of Y" in an entirely constructive sense – you have to actually produce a proof from the axioms.

Posted by: Walt on January 14, 2013 11:11 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Well, what you said in your previous comment was a r.e. list of theorems. I assumed you meant r.e. in the usual sense, which requires a mathematical abstraction of a computer. Most theorems in that list have more symbols than there are atoms in the observable universe, so there's not much hope of a *physical* computer generating the whole list. But once you're in the realm of a mathematical abstraction, then the list that you get depends on the assumptions you make – the list might contain $0 = 1$ or not depending on what world of mathematics you choose in which to discuss the abstraction.

Posted by: Mike Shulman on January 14, 2013 4:16 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Then whatever would be the least controversial choice. Peano Arithmetic?

I didn't expect argument from this direction, actually. Ultrafinitists reject the notion that really large integers are meaningful, but I think most mathematicians believe that there's an objective truth to the matter of whether Ackermann(unimaginably huge number, unimaginable huge number) has 107 prime factors has an objective answer, independent from choice of foundations, even if we know of no formal system that will allow us to answer the question in a reasonable amount of time.

Posted by: Walt on January 14, 2013 5:31 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

And in what meta-metatheory do you wish to discuss the question of which formulas are asserted by Peano Arithmetic to be theorems of ZFC? (-: If ZFC is inconsistent, then there is a very large integer which codes for a proof of $0 = 1$ from the ZFC axioms, and PA is sufficient to verify that fact. If ZFC is consistent, then no such integer exists.

It's true that many mathematicians seem to have an ontological committment to the existence of the "true" natural numbers. Post-Gödel, I've never been able to figure out why.

But even if you do believe in the "true" natural numbers, I think the fact that you have to retreat to "a least controversial choice" means that there is no absolute meaning to something like "the theorems of ZFC". We might choose by convention a particular meaning to assign to it, but I think part of Andrej's point is that by trying to enforce any such choice, we close off to ourselves other potential worlds of mathematics.

Posted by: Mike Shulman on January 14, 2013 6:06 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I'm not really retreating to PA. I thought maybe your objection was that I was trying to slip ZFC in as a metatheory. I didn't realize your idea was that "theorem of formal system" doesn't have an absolute meaning, since I would have said that is precisely what formal reasoning gives us. PA $\vdash$ 1+1 = 2. PA $\vdash$ x + y = y + x, etc. We know these things because can produce proof witnesses. No metatheory needed.

The only way you're at the mercy of your metatheory is if you want to prove that something *isn't* a theorem of your formal system. But that is by necessity a statement not just about the original system, but the original system plus the system you are embedding it into. So all you can say is ZFC $\vdash$ "0 = 1 is not a theorem of PA". There's no way to unconditionally prove $0 = 1$ is not a theorem of PA.

Posted by: Walt on January 14, 2013 11:28 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> We know these things because can produce proof witnesses. No metatheory needed.

Agreed. For *particular* theorems with *particular* explicit proofs, we can give an absolute meaning to being a theorem of a formal system.

> The only way you're at the mercy of your metatheory is if you want to prove that something isn't a theorem of your formal system.

Well, you might also want to prove that something is a theorem *without* exhibiting an explicit proof witness.

What I claim is that there's no way to talk about the collection of *all* theorems in some formal system without a metatheory. There's no metatheory-independent "set of all explicit natural numbers". (-:

Posted by: Mike Shulman on January 15, 2013 7:19 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Then we agree, I think. Language is hard.

Posted by: Walt on January 15, 2013 12:23 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Great! It sounded to me like **up here** [http://golem.ph.utexas.edu/category/2013/01/from_set_theory_to_type_theory.html#c043198] you were saying that you wanted to view a concept like "the reals in ZFC" as the "unique absolute object" determined by the collection of all theorems provable about the reals in ZFC. So I was saying that that collection is not determined until you fix a metatheory, hence there is no "unique absolute object" that you can regard as being "the reals in ZFC".

Posted by: Mike Shulman on January 15, 2013 7:11 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

That's why language is hard. I really would call "theorems of a formal system" a "unique absolute object" because there's an explicit function Proof(X, Y) that tells me definitely whether X is a proof of Y. It sounds like for you "unique absolute object" would mean that there exists a function Theorem(Y) that tells you whether or not something is a theorem, which in general is too much to ask for.

Posted by: Walt on January 15, 2013 9:47 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I'm saying that in most cases, there is no meaningful notion of "unique absolute object". A single, particular, explicit theorem of a formal system can be considered so, but once you start talking about "all theorems" then you are sunk. I think this is exactly what you just said, but it seems to me that it implies that there is no "unique absolute object" that can characterize "the reals in ZFC” by “all the theorems that are true about them”, which is what I thought you were proposing.

However, I feel like this argument is starting to repeat itself, so perhaps we should have pity on the bystanders and desist. (-:

Posted by: Mike Shulman on January 15, 2013 10:10 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

We agree on all mathematical points, but it comes down to what we each mean by “unique absolute object”, on which we do disagree. Of course we have *reasons* why we each think what we think, but when we reach the point where my explanations for my position and your objections to my position coincide, then pretty much that’s as far as you can go. My conception of “unique absolute object” includes “recursively enumerable sets”, while yours does not. Every objection you would make to my position I would concede, because they would be different formulations of “but recursively enumerable sets are not recursive”. I agree, recursively enumerable sets are not recursive. On the question of the correct meaning of “unique absolute object”, I can’t imagine a way to make progress. (It wasn’t clear to me that your objection was on precisely this point, so from my point of view the discussion was worth having.)

Posted by: Walt on January 16, 2013 9:53 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Well, I’m still confused. It sure doesn’t look to me as though my objections have to do with r.e. versus recursive. Are you perhaps using the phrase “recursively enumerable set” to mean what I would call a *program* which generates the list of elements of that set? Usually, a “set” is an extensional object which is determined by its elements, so the fact that the same program, interpreted in different metatheories, generates different lists, means to me that there are different “r.e. sets” in different metatheories. But if you’re thinking about the program rather than the output it produces, then I can agree that it’s an absolute object.

Posted by: Mike Shulman on January 16, 2013 5:57 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I’m still confusing. :-) I would say the different metatheories “think” that they prove different theorems, but the actual list of theorems are the ones you get by translating the proofs. So if you have a translation function, Trans that translates theorems and proofs, and a proof predicate for the original theory, Proof1, and one for the metatheory, Proof2, then the recursively enumerable list of theorems are the ones that satisfy Proof2(Trans(X), Trans(Y)), which are just Trans(X), for X in the original list. So if I wrote a program to generate X, then I automatically have a program to generate Trans(X). It won’t be the same set, and Trans might destroy distinctions that existed in the original theory, but it won’t be *bigger*.

Posted by: Walt on January 16, 2013 6:49 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

But it might be the case that $\mathrm{Proof}2(X', \mathrm{Trans}(Y))$ for some X' that is not of the form $\mathrm{Trans}(X)$ for any X. So there might be some Y which is a theorem of the second system (or, I suppose, its translation into the second system is a theorem of the second system) but which is not a theorem of the first system.

Posted by: Mike Shulman on January 16, 2013 7:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Right. That's when the second system "thinks" something is a theorem, but isn't. But the only theorems that are really theorems of the first system are the ones that have proofs in the first system, all of which translate to the second system. So the second system can prove a superset of the theorems of the first system. But at the level of recursively enumerable sets, the theorems of the first system translate into a recursively enumerable set in the language of the second system.

Posted by: Walt on January 17, 2013 10:13 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Can you please be more precise about what your 'first system' and 'second system' are?

Posted by: Mike Shulman on January 21, 2013 6:29 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Sorry, I missed this until now. I'm not quite sure what you're asking.

Posted by: Walt on January 27, 2013 11:18 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

A couple comments ago you started talking about translating from one system into another. Could you specify exactly what these two systems are (in the relevant example) so that I know how to interpret them? One reason I'm confused is that your first comment called these two systems "the original theory" and "the metatheory". But usually I don't think of "translating" something in an object theory into something in a metatheory. The things we say in the metatheory are statements *about* the object theory, not statements that we might also have made *in* the object theory.

Posted by: Mike Shulman on January 28, 2013 12:40 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

In terms of the original topic of the discussion, isn't that exactly what we do? In ZFC, you don't directly prove that $0 \neq 1$ is not a theorem of PA. You write down a model of PA, and prove that in that model that $0 \neq 1$. The translation is the interpretation of the symbols of PA as sets in ZFC.

Posted by: Walt on January 30, 2013 9:17 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Thanks Andrej! I don't know whether I'm a pluralist or a relativist. I used to think that first-order logic was a grand unifying theory. Then I learned about type theory and realized that that didn't fit. But what about **formal systems** [http://ncatlab.org/nlab/show/formal+system] ?

Posted by: Mike Shulman on January 13, 2013 5:26 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Mike's providing us with an exciting blend of considerations in favour of type theory as (i) satisfying existing requirements

> we ought to seek a foundational system which,

and yet (ii) taking us somewhere new

> you *have* to learn to think about it differently.

This allows for a story to be told of how previous systems, material and structural, were as successful as they were, while pointing us to the next chapter of the story.

For those who haven't followed things here, this stretches beyond mere mathematics. Urs is doing great things with type theory and physics, see e.g. **here** [http://golem.ph.utexas.edu/category/2012/09/general_covariance_in_homotopy.html] and **here** [http://ncatlab.org/schreiber/show/Quantum+gauge+field+theory+in+Cohesive+homotopy+type+theory] .

Posted by: David Corfield on January 9, 2013 11:46 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

That's a nice way of putting it. I could see someone else concluding from the same observations that I'm confused or inconsistent. (-:

In fact, it's really just that I'm still just figuring out the best way to present type theory to mathematicians. Perhaps I will always be still figuring that out. Maybe there is no one best way, either, so we just have to present it in lots of different ways and hope that everyone can find something that resonates with them. There are so many aspects to type theory, and I've been moderately surprised to discover recently that sometimes even type theorists can't agree on basic questions. On the plus side, that means that there's more of a chance that any person can find *something* that makes sense to them, if they are open-mindend enough to stick it out until they find it; but on the minus side, it seems that anything we say is also likely to be offputting to *someone*. (-:

Posted by: Mike Shulman on January 10, 2013 4:47 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Mike, this is fantastic and pitched perfectly. Thank you.

Posted by: Emily Riehl on January 9, 2013 10:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

Since we are talking about presenting type theory to a general mathematical audience I have a request in this spirit.

The underlying logic of type theory (I know it's not really "underlying", it's built in on an equal footing with everything else) is by default intuitionistic. However, most mathematics is being done classically. So what is a good way of making logic of type theory classical?

I don't want to discuss merits of doing mathematics constructively. I'm merely pointing out that if type theory aspires to capture everyday practice of mathematicians, then it should make it easy and natural to work classically. I know some one sentence answers to my question, but I'd love to hear some discussion of possible approaches and how well they fit with the core of type theory.

Posted by: Karol Szumiło on January 11, 2013 1:24 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I think I only know of one approach:

```
Axiom lem : forall P:hProp, P or (not P).
```

It doesn't "fit with the core of type theory" in the sense that it destroys the computational aspect, since your "proofs" will get stuck on the axiom when you try to execute them. And of course it rules out lots of interesting categorical models. But if you don't care about that, then it works fine. I think a good number of people using Coq in practice are happy with this axiom. (It does have to be restricted to hprops, otherwise it contradicts univalence; thus it seems especially odd from a proof-relevant point of view.)

Posted by: Mike Shulman on January 11, 2013 7:20 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

That's one of the possibilities I had in mind. (By the way, if we want AC do we just assume an axiom saying that every surjection of h-sets has a section?)

The second approach I was thinking of is "$\lambda\mu$-calculus". The only reference I know is Section 3.2 of Sørensen, Urzyczyn *Lectures on the Curry-Howard Isomorphism*. I don't remember very well how it works, but it basically adds a new type of terms (μ-abstraction) that handle arguments by contradiction. I like this idea because it treats the law of excluded middle on an equal footing with the rest of type theory while the first approach just throws it in as an additional assumption that stands out as something odd. There is of course a question whether it has some reasonable interpretation for general types of HoTT.

Posted by: Karol Szumiło on January 11, 2013 8:13 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Another option is to work with algebras for the double-negation monad. BTW, is there a nice characterization of these in HoTT? Certainly anything of the form $A \to \bot$, and decidable types are algebras. Anything else?

Regarding the $\lambda\mu$-calculus and variations; I think that's a very appealing idea, especially if it could restore the symmetry of fibrations and cofibrations, and perhaps make (higher) coinductive types more perspicuous. But as far as I know, all non-trivial models are lop-sided one way or the other (like the models for Call-by-Push-Value, which is also relevant).

Posted by: Ulrik Buchholtz on January 11, 2013 10:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Mike gives one approach by adding classical axioms. There are other approaches as well.
Another approach is to add control operators to the term language of a type theory. These allow one to prove classical results. There have been a lot of work on this area. It is still being studied today. Here is a non-exhastive list of classical type theories.

First, it all started with Griffen's realization that Felleisen's $\mathcal{C}$-operator could be typed with the law of double negation. However, subject reduction failed. Much work followed to fix this.

Later, Parigot invented the $\lambda\mu$-calculus which is an extension of the simply typed λ-calculus. It adds the μ-operator which is a control operator. It is straightforward to prove the law of excluded middle. For a proof see **this** [http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDUQFjAA&url=http%3A%2F%2Fcs.ru.nl%2F~freek%2Fcourses%2Ftt-2011%2Fpapers%2Fparigot.pdf&ei=rJDwUIbvCpTM9ATl-YGICQ&usg=AFQjCNFYIJku7n68Kkoi7hdngeNj9-SVxA&sig2=mwcRKzBfxhRLG2z-CaVhMw&bvm=bv.1357700187,d.eWU] .

Rehof and Sorensen define the $\lambda\Delta$-calculus which adds a control operator whose type is the law of double negation. The language enjoys subject reduction. See **this** [http://ls14-www.cs.tu-dortmund.de/images/7/76/TheLambdaDelta.pdf] .

Using the $\lambda\Delta$-calculus Sorensen et al. give the classical λ-cube in **this** [http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=79BAEC2D243E548A793DF535AB22FB0A?doi=10.1.1.32.1371&rep=rep1&type=pdf] . This is the only know classical dependent type theory with control.

Later, Curien and Herbelin define a really cool type theory called the $\bar{\lambda}\mu\tilde{\mu}$-calculus. This makes the dualities of the classical sequent calculus LK explicit. They use the dual of implication called subtraction. Subtraction is due to Crolard's work. See **this** [http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDUQFjAA&url=http%3A%2F%2Fpauillac.inria.fr%2F~herbelin%2Fpublis%2Ficfp-CuHer00-duality.ps.gz&ei=fZLwULDjBZKu8QSzxoGADQ&usg=AFQjCNHBKyUWxLYJZAJHM33pWi73yJPlxA&sig2=JJ3_zeSSVovyTB5K9vTU4Q&bvm=bv.1357 and **that** [http://cedric.cnam.fr/~crolardt/publications/tcs.pdf] .

Wadler didn't care for subtraction so he defined what he calls the dual calculus. Which does not have implication, but it can be defined. See **this** [http://homepages.inf.ed.ac.uk/wadler/topics/dual.html] .

Both Curien and Herbelin's and Wadler's type theories were used to prove that call-by-name is dual to call-by-value.

There is a lot of other work on things like delimited control. I do not list everything here, but those listed above are some of the highlights of the body of research. There is a lot of work still needing done.

One main problem with control operators is that canonicity does not hold. This is the part of the focus of a lot of ongoing work such as my own. In addition notice that all of the above work is on simply typed calculi. Extensions to dependent types is still an open problem.

I am really interested in knowing how these control operators fit into HOTT.

I hope this helps.

Harley

Posted by: Harley Eades on January 12, 2013 1:38 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Thanks for all the suggestions! I realized after I wrote my last comment that I also know another way, also involving double negation. In type theory, any (idempotent, monadic) modality gives rise to a corresponding "logic" by reflecting all statements into the modal types as necessary. For instance, if your modality is called $\diamond$, and A and B are modal types (i.e. the canonical maps $A \to \diamond A$ and $B \to \diamond B$ are equivalences), then "A $\diamond$-and B" means $A \times B$, since this is automatically modal, while "A $\diamond$-or B" means $\diamond(A + B)$. When $\diamond$ is the identity (all types are modal), then you get propositions-as-types logic, while when $\diamond$ is the hprop-reflection ((-1)-truncation), you get the hprop-logic. One may argue that the best solution to the problem of language ("which types are propositions?") is to work parametrically over modalities as much as possible, in the same way that combinators in Haskell tend to be parametric in a monad. And one such modality, of course, is double negation, $\diamond A \equiv \neg\neg A \equiv (A \to \emptyset) \to \emptyset$. (Computationally, this is like using a continuation-passing style.) Note that $\neg\neg A$ is always an hprop, so the double negation logic is coarser than the hprop logic.

Using type theory with double-negation-logic is "classical" in a certain sense. For instance, LEM holds, in that for all A we have "A $\neg\neg$-or $\neg A$". However, it's not exactly the same as what you would get by assuming LEM as an axiom for all hprops. For instance, it doesn't satisfy the law of unique choice: if for every $a : A$ there $\neg\neg$-exists a unique $b : B$ such that $P(a, b)$, it doesn't follow that there $\neg\neg$-exists a function $f : A \to B$ such that $P(a, f(a))$ for all $a : A$. Both the identity modality and the hprop modality do satisfy this principle.

In an extensional, impredicative world, the fix for this is to restrict consideration to those types which are "$\neg\neg$-sheaves", i.e. right orthogonal to all $\neg\neg$-dense inclusions. The $\neg\neg$-sheafification is another modality, indeed a lex modality (= subtopos), and by working only with types in that modality we have a very classical behavior. Is this what you meant by the "double-negation monad", Ulrik? However, I don't know whether there is a homotopical and/or predicative version of $\neg\neg$-sheaves. (This came up recently in **another discussion** [http://homotopytypetheory.org/2012/11/19/all-modalities-are-hits/] .)

These other calculi sound very interesting; I will have to learn about them!

(For AC, yes, we can assert that surjections of hsets have sections, where "surjection" is interpreted in the hprop sense. It's not really clear whether there is a consistent and useful extension of AC which says anything about non-hsets.)

Posted by: Mike Shulman on January 12, 2013 6:36 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> Is this what you meant by the "double-negation monad", Ulrik?

Actually, I just meant the double-negation endofunctor but wrote monad (the monad-algebras are rarer, I guess).

But you bring up some great points regarding predicativity: for double negation to work internally as continuation-passing style, it should be defined as $\mathrm{dn}(A) := \Pi_{X:\mathrm{Type}}((A \to X) \to X)$, but that lands in the next universe.

Posted by: Ulrik Buchholtz on January 12, 2013 8:06 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I don't think the excluded middle is definable as continuations with a polymorphic answer type, since one level up you can define the function:

$$\begin{aligned} f &: \mathrm{dn}(A) \to A \\ f &= \lambda d : \mathrm{dn}(A).\ d\ A\ (\lambda a : A.\ a) \end{aligned}$$

Indeed, in System F you can show that this is one half of an isomorphism between A and $\forall X.\ (A \to X) \to X$. I've always thought of this isomorphism looks morally like an instance of the Yoneda lemma, with $\mathrm{Nat}(\mathrm{Hom}(A, -), \mathrm{Id}) \simeq A$, and so predicatively the jump in the universe level is more or less what you'd expect.

I think CPS with a fixed answer type $\diamond A \triangleq (A \to 0) \to 0$ looks more like double-negation (and it doesn't kick you up a universe level, either).

Posted by: Neel Krishnaswami on January 12, 2013 12:10 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Yes, that's all true.

I was thinking about when we want to instantiate a particular answer type internally, but not decide on any particular one in advance, then we need a quantifier somewhere.

Now I remember that I have actually seen a solution before: indexed monads (see http://mattam.org/repos/coq/misc/shiftreset/GenuineShiftReset.html and references therein).

Posted by: Ulrik Buchholtz on January 12, 2013 8:06 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Not to be confused with what a category theorist calls "indexed monads" (i.e. monads in the 2-category of **indexed categories** [http://ncatlab.org/nlab/show/indexed+category]).

Posted by: Mike Shulman on January 13, 2013 5:27 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

It is definitely almost an instance of the Yoneda lemma. But I thought you needed to add a naturality condition in order to get an isomorphism. (Steve Awodey and Andrej Bauer have been working on this; Andrej gave a talk about it at IAS last year, and the big deal was how to incorporate naturality and coherence for that naturality for types of higher h-level.) In fact, how do you even formulate "isomorphism" in a theory like system F without identity types?

Posted by: Mike Shulman on January 13, 2013 5:23 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I am not a homotopy type theorist, but I am curious about this stuff. I've seen a couple of talks by Voevodsky that he gave at the IAS and I noticed and have read that a computer proof assistant like Coq is often used to verify proofs.

I have seen some of the statements (such as those above) from HoTT, and they resemble formal statements of first-order logic and such, and seemingly written for machine processing.

Is a computer necessary to do mathematics in HoTT? Or is it possible to write out the statements of HoTT in natural sentences and reason "by hand" as in the mathematics of today?

Although I've read several pages about type theory I am still a bit confused on how necessary computer verification would be in proving theorems with fairly long proofs.

Posted by: Jason Polak on January 11, 2013 7:48 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

A very interesting and important question! So far, most mathematics in HoTT has been done in a computer-assisted way. However, one of the ongoing projects at IAS this year (initiated by Peter Aczel), which I've mentioned **before** [http://golem.ph.utexas.edu/category/2012/11/freedom_from_logic.html] , is to develop a style (or styles) of doing "informal" mathematics in HoTT as well. The idea is that this would be a way of writing mathematics which stands in the same relation to computer-formalized HoTT that everyday written mathematics nowadays stands to (say) ZFC. I think we've made good progress on this, and I'll hopefully there'll be some good stuff to share soon.

Posted by: Mike Shulman on January 12, 2013 6:40 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

I enjoyed this description of type theory. It made me feel I might understand this subject some day.

I wanted to ask a question, which I hope is not off-topic.

My main interest in foundational studies is not that I am concerned about inconsistencies arising in mathematics. I would like to understand applications to computer science and specifically to the design of systems for computer algebra and combinatorics.

1. I have read various postings about computer science, automated theorem proving, lambda calculus. None of this, unless I missed something, covers object oriented languages. What is the logic of object oriented languages?

2. sage-combinat, following Magma, has Categories, Parents, Elements which according to the documentation is an implementation of concrete categories. This suggests to my naive mind, that it could be useful to have a foundation for these. I expect that the list of axioms would be longer than the alternatives already discussed here. I don't see this as, per se, a drawback. It seems that what causes headaches is implementing coercions.

Posted by: Bruce Westbury on January 14, 2013 7:07 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> My main interest in foundational studies is not that I am concerned about inconsistencies arising in mathematics.

I think that's true for practically everyone. I can only think of one or two people that I've ever heard of who might seriously believe that there may be inconsistencies in mathematics.

> What is the logic of object oriented languages?

At a basic level, objects are just Sigma-types with some syntactic sugar. When you get into fancier stuff then it gets more complicated. You may be interested in **this blog post** [http://existentialtype.wordpress.com/2012/08/25/polarity-in-type-theory/] by Bob Harper, and Part IX of his book *Practical Foundations for Programming Languages*.

> it could be useful to have a foundation for these.

You mean, a foundation for concrete categories? Concrete categories are easily definable in pretty much any foundation; I'm not sure what you're envisioning.

Posted by: Mike Shulman on January 14, 2013 8:51 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

If we agree that potential inconsistencies are not the issue then can I clarify what are the issues? As I understand it there are two reasons for getting involved in this discussion. One is pedagogical; the aim here is to give some foundations that can be presented to students. This is laudable and I think I understand criteria for judging the various proposals. The other reason is to try and capture what it is that mathematicians do. I am much less clear about how to assess different proposals and so I find this more intriguing.

Let's concentrate on the issue of trying to encapsulate what it is that mathematicians do. The traditional approach is to look in the research literature. This is a top down approach and leads to work on automated theorem proving or automated proof verification. An alternative is to look at contemporary computer systems which now incorporate significant parts of mathematical knowledge. This is more of a bottom up approach. My feeling is that all of the current computer algebra systems are struggling to scale up.

My vision for concrete categories comes from a position of ignorance. I accept that all reasonable foundations allow an implementation of concrete categories. You added “easy” which is subjective. What I had in mind was that if I was asked to write out the definition of a concrete category then I would use set several times and with several meanings. I think I would regard the set of objects as a material set and each hom-set as a structural set. This would then lead to a foundation which would incorporate both aspects. This would be unsatisfactory from a pedagogical perspective as it would be unnecessarily complicated. However this may be closer to what working mathematicians are currently doing with large scale software.

Posted by: Bruce Westbury on January 15, 2013 12:20 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

One of the main points of this post is that type theory is the natural way to incorporate material-sets and structural-sets into a single foundation. I don’t find it unnecessarily complicated, although of course that is also subjective. But it’s quite simple to define concrete categories in type theory, in such a way that the objects and morphisms carry the appropriate “internal structure” of concreteness, but no more. For instance, the category of groups has type of objects given by

$$\mathrm{Grp}_0 \coloneqq \sum_{G:\mathrm{Type}} \sum_{e:G} \sum_{m:G\times G\to G} \mathrm{GroupAxioms}(G, e, m)$$

and types of arrows given by

$$\mathrm{Hom}_{\mathrm{Grp}}((G, e, m, \ldots), (G', e', m', \ldots)) \coloneqq \sum_{\phi:G\to G'} \Big((\phi(e) = e') \times \prod_{x,y:G} (\phi(m(x, y)) = m'(\phi(x), \phi(y))) \Big).$$

Thus, Grp_0 is “material” in the sense that its elements have internal structure: they are tuples consisting of a type and a group structure on that type. $\mathrm{Hom}_{\mathrm{Grp}}$ is “material” in the same way: its elements are actual functions together with witnesses of the homomorphism property. (Why do you say you would make the hom-sets structural? The morphisms in concrete categories are generally just as concrete as the objects.) On the other hand, both are “structural” in the sense that belonging to them is a typing judgment rather than a proposition.

I don’t really know much about how modern computer algebra systems are implemented. But I do know something about computer proof assistants (which seem at least as important from a “what mathematicians do” viewpoint), and many of the most successful of those are indeed built with type theories.

Posted by: Mike Shulman on January 15, 2013 7:51 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

As with Tom’s post, this interests me personally mostly from a pedagogical perspective. **As Eugene noted there** [http://golem.ph.utexas.edu/category/2012/12/rethinking_set_theory.html#c043129] , it’s nice to know that it’s okay to introduce functions as primitives. The take-away message I find here is that it’s similarly okay to tell students that statements like “$x \in f$” or “$x \cap A$” don’t even make sense to think about when x is a real number, f is a function, and $A \subseteq \mathbb{R}$.

I’m thinking here of students in, say, a first course in linear algebra, who may need to be reminded that you can’t add a scalar to a vector. Sometimes a savvier student in the same class has been introduced to the idea that “everything is a set” and objects on that basis that therefore “$x \in f$” and “$x \cap A$” are perfectly meaningful, even if they’re irrelevant to the present class. So rather than say, “Yes, that’s true, but it’s better here to ignore that fact,” I can respond more informatively with “That may or may not be true depending on our foundations, and since we don’t want the answers to linear algebra questions to depend on such matters, it’s better that we take such statements to be meaningless.”

Posted by: Mark Meckes on January 14, 2013 7:58 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

A nice point! This is a good counterargument to the ZFC-ist claim that “yes, there are these dumb things that ZFC technically lets you do, but no one would be silly enough to do them.” Maybe no practicing mathematician is that silly, but students routinely make type errors, and it’s useful to have a foundational system that shows them for what they are.

Posted by: Mike Shulman on January 14, 2013 8:58 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

These two threads have persuaded me that there is something wrong with how the “everything is a set” idea gets taught, in a way that actually is damaging. The idea of axiomatic set theory is to write down a system with as few primitives as possible that we believe to be consistent, and can represent mathematics. Not only is function not a primitive of ZFC, *equality* isn’t even a primitive. (It’s definable in terms of ϵ.) If people come away with the notion that you can’t take functions as primitives, that’s bad. You can take all kinds of things as primitives, if you want to.

There’s no ZFC-ist objection to the idea that expressions naturally have types, or even to the idea of type theory. The objection is to the idea that you *must* use types, which commit you to a tightly-controlled set of formulas. Set theory can represent typing information as unary predicates, which then can be used in conjunction with ordinary logical connectives. For example, the sentence “x is a scalar or x is a vector” seems perfectly intelligible, as is “x is either a two-element set, or a function from Z to R”. Work in type theory has shown that you don’t *need* the ability to say this to prove the bulk of theorems of mathematics, but intuitively I would find it odd if I couldn’t.

Posted by: Walt on January 15, 2013 12:09 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Based on what you say, the analogy

> set theory : type theory :: assembly language : high-level language

seems very close. Assembly language has a very small number of primitives which are easy to implement and debug in a processor, but which are sufficient to represent all programs. It doesn’t prevent you from using a high-level typed language, but when writing in assembly itself you are free to do things like multiply together two memory addresses. In set theory, everything is a set; in assembly language, everything is a bit sequence.

I think this analogy is also useful for understanding the advantages of type theory. Yes, in assembly language you can carry around typing information by hand, or just remember that register AX is currently a memory address while register BX is an ASCII code. But it’s much better to use a programming language which does typechecking for you, so that your type errors get caught automatically at compile time. You give up the ability to say some things that you never (or rarely) want to say in practice anyway, in exchange for being prevented from mistakenly saying a lot of things that you often say by accident but don’t intend.

On the other hand, this analogy breaks down under further inspection, because as I argued in the main post, set theory actually contains no fewer primitives than type theory does. Membership $\in$ is not the only primitive of set theory: there are also $\wedge$, $\vee$, $\forall$, $\exists$, etc. coming from the framework of first-order logic on which set theory is built. Type theory uses these same logical primitives, extended to operations on types rather than merely operations on propositions. So even if your goal is to reduce mathematics to the fewest primitives possible, type theory is just as good as set theory.

Posted by: Mike Shulman on January 15, 2013 7:37 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Material set theory with classical first-order logic requires only four primitives: $\bot$, $\Rightarrow$, $\forall$, and $\in$. Everything else can be encoded using these.

Posted by: Toby Bartels on January 15, 2013 8:25 AM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Three, actually. You can merge $\bot$ and $\Rightarrow$ into the Sheffer stroke $P \uparrow Q$, (better known these days as NAND).

Of course, only a barbarian would regard such a mutilation of logic as an improvement.

Posted by: Neel Krishnaswami on January 15, 2013 2:00 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

It always bothers me a little bit when I read that Boolean algebras can be defined by a single binary operation such as NAND, since this would seem to allow that a Boolean algebra can be empty. In my book, a Boolean algebra always has constants $\bot$, $\top$ (possibly equal).

(At one point in the past, one never considered empty domains, or at least that was considered “ungentlemanly” as it were. But that attitude definitely seems to belong to the past.)

Posted by: Todd Trimble on January 15, 2013 3:04 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I think that’s directly the result of category theory. If you have a signature with no symbols, then the empty set is the initial object.

The reason for rejecting the empty carrier before was that a bunch of tautologies of first-order classical logic implicitly assume a non-empty carrier. For example, $\forall x A \Rightarrow \exists x A$ is only a tautology for non-empty sets. (I think something similar holds for intuitionistic logic, but I know less about that.)

Posted by: Walt on January 15, 2013 3:36 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Categorical logic had something to do with recognition of the *correct handling* of empty domains in logic, both intuitionistic and classical, but generally speaking, recognition and acceptance of empty structures in mathematics is something much more broadly cultural. The statement I referred to has nothing particularly to do with category theory; it’s an assertion made about the variety of Boolean algebras that you can find in the literature, and it’s wrong according to the universally accepted definition of model of a theory: if there are no constants in the theory, then the empty set carries a (unique) model structure.

Posted by: Todd Trimble on January 15, 2013 4:17 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

In general, I agree with Todd. (I've never understood why you would take "$\forall x A \Rightarrow \exists x A$ is a tautology only for non-empty sets" as a reason for rejecting *empty sets* rather than simply realizing that $\forall x A \Rightarrow \exists x A$ is *not* a tautology.)

However, in the case in question (material set theory in classical FOL), the other two operations allow us to define some closed formula, like $\forall x, x \in x$. We can then define the formula $\bot$ using NAND in the usual way. (I have to agree with Neel, though, that defining $\bot$ to be $(\forall x, x \in x) \uparrow (\forall x, x \in x)$ is a mutilation of logic.)

Posted by: Mike Shulman on January 15, 2013 6:58 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

The issue is even more pronounced and urgent when we extrapolate further to reasoning inside a topos (for example), where such classical "tautologies" fail over any domain X which is not well-supported (meaning the map $X \to 1$ is not a regular epi). As can happen all the time, as in the example of sheaves over a space.

While it doesn't refute the exact words of my comment, I'll freely admit that $\top \coloneqq \forall x\ (x \in x)$ hadn't occurred to me. And for some reason, I am not ashamed to admit that. :-)

Posted by: Todd Trimble on January 15, 2013 7:19 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Tautology is a function of the logical system. If $\exists x.\ x = x$ is an axiom of your system, it's an axiom of your system, just like the law of excluded middle. I agree that a minimal logic shouldn't impose it.

The real reason for imposing it in classical FOL is that the rules for converting a formula to prenex normal form are not valid for empty domains. Since analyzing formulas in terms of their alternating qualifiers is a central technique in classical predicate logic, people just throw out the empty domain rather than handle it as a special case. I don't know what the argument for it is in intuitionistic logic. It seems like it wouldn't cost you anything there.

Back when I was a high school student, and therefore prone to make impulsive decisions, I tried to read *Principia Mathematica*. The local library had the second edition, which contains a section on how wonderful Sheffer's invention of the Sheffer stroke was, and how everything should be formulated in terms of it. It was a long time before I looked at formal logic again…

Posted by: Walt on January 15, 2013 8:45 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

> The real reason for imposing it in classical FOL is that the rules for converting a formula to prenex normal form are not valid for empty domains.

Okay, that's a fair point.

Posted by: Mike Shulman on January 15, 2013 9:12 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Yes, and as I mentioned in the post, type theory can be presented with only three primitives: universes, inductive types, and dependent products. But competing for who has the fewest primitives (can anyone get down to two? (-:) is not really the point.

The main thing I meant to point out is that thinking of “everything is a set” as a very successful reduction to a single primitive is a bit of an illusion, because the whole structure of FOL is hiding in the background (whatever primitives you choose to present it in terms of). Now I will concede that for pedagogical purposes, it may make students happy and comfortable to be told that “everything is a set”: since they have been speaking in first-order logic their whole lives without realizing it, they won’t see the substantial structure underpinning “everything is a set”. However, I think the advantages of teaching students to think in a typed way (among which is the fact that you can simultaneously be teaching them to write computer programs, prove theorems, prove theorems about their programs, and write programs to implement their theorems) outweigh the advantages of giving them this somewhat-illusory warm and fuzzy feeling. (-:

Posted by: Mike Shulman on January 15, 2013 7:06 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

An article from the latest issue of *AMS Notices* is tangentially relevant: Raymond T. Boute, **How to Calculate Proofs: Bridging the Cultural Divide (pdf link)** [http://www.ams.org/notices/201302/rnoti-p173.pdf] . The article is worth reading for its emphasis on rigorous logical reasoning aided by symbolic manipulation (similar to the style that inteactive proof assistants naturally support).

Unfortunately, Boute advocates for a ‘naïve set-theoretical’ formalism which includes what may be considered to be severe drawbacks. For instance, his notation uses a function abstraction operator $v : S \cdot\wedge\cdot\, p\ .\ e$ which combines ordinary lambda abstraction with subsetting the domain to the set which satisfies predicate $p(v)$. Then the authors define curly brackets { } as the image of a function and v: S ∣ p as the constant function on S restricted to the subset of S satisfying $p(v)$. They do this in order to recover a sigma-like operator {v:S ∣ p}. The “point-free” definitions of ∀ and ∃ are perhaps even more confusing.

I see these foundational issues as fairly severe drawbacks in an otherwise very good article. Since Boute’s proposal is intended to improve mathematical education in a very practical sense (including to some extent in high school math), I’m thinking that someone with expertise in conceptual foundations should write to *Notices*, pointing out these confusing issues in the article, and perhaps exploring some kind of simplified type theory as a potential solution to these concerns.

Posted by: John Smith on January 15, 2013 5:20 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

Mike, brilliant post! Many thanks. I had that elusive feeling of “oh, yeah, I knew that, this is how it is!”. Then got totally bored and confused on the discussion after. what about summarizing main questions and answers in a new post?

Posted by: Valeria de Paiva on January 25, 2013 3:00 PM | Permalink | Reply to this

Re: From Set Theory to Type Theory

[http://golem.ph.utexas.edu/~distler/blog/mathml.html]

I’m not sure I think any of the discussions are really worth trying to summarize. I hope they were helpful for the people involved (they were for me), but if they weren’t interesting or didn’t make sense to you, then I’m a little doubtful that a summary would be any different. It might be worth reading **this comment** [http://golem.ph.utexas.edu/category/2013/01/from_set_theory_to_type_theory.html#c043063] , I guess, in which I summarized some of the ideas I’d come to from the foregoing discussion.

Posted by: Mike Shulman on January 26, 2013 7:48 PM | Permalink | Reply to this

Read the post **Category Theory in Homotopy Type Theory**
[http://golem.ph.utexas.edu/category/2013/03/category_theory_in_homotopy_ty.html]
Weblog: The n-Category Café
Excerpt: Doing category theory in homotopy type theory allows us to satisfy the principle of equivalence

automatically.

Tracked: March 5, 2013 4:20 AM

Post a New Comment

view chronologically