Blog Post

Nuiok and Paul

Way too late...

1 Introduction

Many well-known type theories, e.g. Martin-Löf dependent type theory or linear type theory, have as their starting points particular forms of reasoning that we wish to treat syntactically. In a sense, it is only a happy accident that these type theories then provide means to reason *about* locally cartesian closed categories or $\star$-autonomous categories.

Mike Shulman's

"A Practical Type Theory for Symmetric Monoidal Categories"

revererses this; he starts with symmetric monoidal categories as the intended semantics and then (reverse)-engineers a syntax in which in which it is *practical* to reason about such categories.

But which properties of type theory make it practical? Shulman, sythesizing various observations, settles on a few basic tenets to guide the invention of the syntax; we reduce these further to the conditions that the type theory be: (1) intuitive, (2) economical, and (3) effective.

intuitive First, a practical type theory should permit us to leverage our intuition with reasoning about 'sets with elements'. What this means in practice is that we require our type theory, and the desired internal-language/term-model relationship with symmetric monoidal categories, to provide one-to-one correspondences:

$$\begin{aligned} \text{objects} &\longleftrightarrow \text{contexts} \\ \text{morphisms} &\longleftrightarrow \text{typing judgements} \\ \text{commuting polygons} &\longleftrightarrow \text{equality judgements} \end{aligned}$$

It is by doing so that we'll be free to think of types as sets, and to think of typed terms as elements in a way that applies to all symmetric monoidal categories.

economical Second, when working with a specific category, one usually works, whether we acknowledge it or not, with a presentation in terms of generators in relations, e.g. $\triangle$ as generated by face and degeneracy maps subject to the simplicial identities. As such, our type theory should not be directly specified in terms of a symmetric monoidal category, but rather in terms of a economical presentation of it.

effective Lastly we ask that, given a presentation for a symmetric monoidal category, the type theory we get therefrom should be complete for that symmetric monoidal category. By this we mean that something should hold in a particular symmetric monoidal category if and only if it is derivable as a judgement in the associated type theory.

2 Symmetric Monoidal Categories and Presentations of PROPs

Shulman builds a practical type theory for each symmetric monoidal from a particularly economical description (of it); Shulman builds a practical type theory for each symmetic monoidal category from *a presentation by free* PROPs.

While it is 'well known' that every symmetric monoidal category is equivalent to a symmetric *strict* monoidal category, it is probably less well known that every symmetric strict monoidal category is equivalent to a PROP.

Definition 1. A PROP, $\mathcal{P}$, consists of a set P of generating objects and a symmetric strict monoidal category whose underlying monoid of objects is the free monoid on the set P.

The equivalence between PROPs and symmetric monoidal categories just unwinds every equality of objects $A \otimes B = C$ into an isomorphism $A \otimes B \xrightarrow{\sim} C$, rendering free the monoidal product.

Example. Given a set X, the **free permutative category on** X, Σ_{X}, whose monoid of objects is the monoid of lists drawn from the set X and whose morphisms are given by the expression

$$\Sigma_{\mathsf{X}}\left(\overrightarrow{X}, \overrightarrow{Y}\right) = \left\{ \sigma \in S_n \;\middle|\; \overrightarrow{X_\sigma} = \overrightarrow{Y} \right\}$$

(where by $\overrightarrow{X_\sigma}$ we mean the reorganization of the list $\overrightarrow{X}$ according to the permutation σ) is a PROP.

Example. For a more complicated example, let X_0 be a set and let

$$\mathsf{X}_1 = \left\{ (\overrightarrow{X}_i, \overrightarrow{Y}_i) \in \mathsf{List}(\mathsf{X}_0)^2 \;\middle|\; i \in I \right\}$$

be a set of pairs of lists valued in X_0. Let $F(\mathsf{X}_1, \mathsf{X}_0)$ denote the free symmetric monoidal category generated by Σ_{X_0} together with additional arrows $f_i : \overrightarrow{X}_i \longrightarrow \overrightarrow{Y}_i$ for each $i \in I$. $F(\mathsf{X}_1, \mathsf{X}_0)$ is also a PROP. (We'll play fast and loose with the form of elements of these sets X_1 later on, failing to distinguish between the name f_i and the element $(\overrightarrow{X_i}, \overrightarrow{Y_i})$)

Importantly, this second example is very nearly general - every PROP admits a bijective on objects and full, but not in general faithful, functor from some PROP of the form $F(\mathsf{X}_1, \mathsf{X}_0)$.

Example. Let X_0 and X_1 be as in the previous example and let

$$\mathsf{R} = \left\{ (s_j, t_j) \in \mathsf{Mor}(F(\mathsf{X}_1, \mathsf{X}_0))^2 \;\middle|\; j \in J \right\}$$

be a set of pairs of morphisms in the PROP $F(\mathsf{X}_1, \mathsf{X}_0)$. Let $F(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$ be the quotient of the symmetric monoidal category $F(\mathsf{X}_1, \mathsf{X}_0)$ by the congruence generated by $R \subset \mathsf{Mor}(F(\mathsf{X}_1, \mathsf{X}_0)) \times \mathsf{Mor}(F(\mathsf{X}_1, \mathsf{X}_0))$.

As you might hope, this last example is fully general. Every PROP, hence every symmetric monoidal category, is equivalent to a PROP of the form $F(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$.

3 From Presentation $(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$ to Type Theory $\mathsf{PTT}_{(\mathsf{R},\mathsf{X}_1,\mathsf{X}_0)}$

It is from these presentations of $\mathsf{PROP}s$, the triples $(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$, that we'll build type theories $\mathsf{PTT}_{(\mathsf{R},\mathsf{X}_1,\mathsf{X}_0)}$ for the symmetric monoidal categories $F(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$. Indeed, reading $\rightsquigarrow$ as "gives rise to"

$$\begin{array}{rcl} \mathsf{X}_0 & \rightsquigarrow & \text{contexts} \\ \mathsf{X}_1 & \rightsquigarrow & \text{typing judgments} \\ \mathsf{R} & \rightsquigarrow & \text{equality judgement} \end{array}$$

3.1 Contexts

The contexts of $\mathsf{PTT}_{(\mathsf{R},\mathsf{X}_1,\mathsf{X}_0)}$ (usually denoted Γ, Δ, etc.) are lists

$$x_1 : A_1, \ldots, x_n : A_n$$

of typed variables, where the A_i are elements of the set X_0. It's not hard to see that, up to the names of variables, contexts are in bijection with $\mathsf{List}(\mathsf{X})$.

3.2 Typing Judgments

Typing judgements, as promised, correspond to morphisms in $F(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$. For example:

$$\begin{array}{rcl}
\text{judgements} & & \text{morphisms} \\
x : A \vdash f(x) : B & \longleftrightarrow & f : (A) \longrightarrow (B) \\
x : A \vdash \big(h_{(1)}(x), h_{(2)}(x)\big) : B & \longleftrightarrow & h : (A) \longrightarrow (B_1, B_2) \\
\vdash (|z^a) : () & \longleftrightarrow & z : () \longrightarrow ()
\end{array}$$

The rules from which these judgments may be derived correspond, roughly speaking, to applying a tensor product of generating morphisms in X_1 composed with a braiding - something like

$$\frac{\begin{array}{c}\Gamma \dashv (\overrightarrow{M}, \ldots, \overrightarrow{N} | \overrightarrow{Z}) : (\overrightarrow{A}, \ldots, \overrightarrow{B}) \\ (f : \overrightarrow{A} \to \overrightarrow{F}) \in \mathsf{X}_1, \ldots, (g : \overrightarrow{B} \to \overrightarrow{G}) \in \mathsf{X}_1 \qquad (\sigma : (\overrightarrow{F}, \ldots, \overrightarrow{G}) \to \triangle) \in \Sigma_{\mathsf{X}_0}\end{array}}{\Gamma \dashv (\sigma(\overrightarrow{f}(\overrightarrow{M}), \ldots, \overrightarrow{g}(\overrightarrow{N}))) : \triangle}$$

in which:

- $\overrightarrow{M}, \ldots, \overrightarrow{N}$ are terms of types $\overrightarrow{A}, \ldots, \overrightarrow{B}$ respectively;
- the f through g are generating arrows in the set X_1; and
- σ is the avatar in Σ_{X_0} of some permutation.

Now, it's pretty clear that we're hiding something - the gorey details of the two rules which derive typing judgments. However the reader should rest assured that:

- not only are the details not that onerous; but
- more naive structural rules corresponding precisely to:

– tensoring;

– composition; and

– braiding

are admissable - one is free to work with these more intuitive rules, and let the careful magic of the type theory do the rest behind the scenes.

3.3 Equality Judgements

Equality judgements, e.g.

judgements commuting polygons

$$x : A \vdash f\left(x\right) = h\left(g\left(x\right)\right) : B \quad \longleftrightarrow \quad \begin{tikzcd} A \arrow[r, "g"] \arrow[dr, "h"'] & (B) \arrow[d, "f"] \\ & (C) \end{tikzcd}$$

are similarly derived from rules coming from the set R. We'll be more explicit in the context of an example later on.

4 Sweedler's notation and Shulman's judgements

A little later on we'll illustrate the type theory with an example - in Section 5 we'll recover Shulman's proof that trace is cyclic in the practical type theory for the free dual pair. Before we do so however we'll expound on what is perhaps the central insight of Shulman's paper.

Until this point it's not actually terribly clear what makes this type theory specifically adapted to symmetric monoidal categories as opposed to cartesian monoidal categories. After all, we've written contexts as lists, precisely how we write contexts in a cartesian type theory.

4.1 Of lists and cartesian/symmetric-monoidal products

While we don't always think about it, we write contexts in cartesian type theories as lists of typed variables because of the universal property of the product - a universal property characterised in terms of projection maps. Indeed, in a cartesian type theory we write contexts as lists because lists can be recovered from the list of their projections. This is not the case for arbitrary symmetric monoidal products however - in general there are no projection maps and even when there are, there is no guarantee that theyve a similar computation rule.

To illustrate this in the context of mathematics, consider a pair of vector spaces U_1 and U_2. Any element $z \in U_1 \times U_2$ is uniquely specified by the pair of elements $\mathsf{pr}_1(z) \in U_1$ and $\mathsf{pr}_2(z) \in U_2$. However, elements w of the tensor product $U_1 \otimes U_2$ need not be simple tensors of the form $x \otimes y$ and, instead, are generally linear combinations $\sum_{i=1}^{k} x_{1,i} \otimes x_{2,i}$ of simple tensors.

Provided we are careful however, we can still use lists - this is what Shulman, following Sweedler does. Consider a general element $\sum_{i=1}^{k} x_{1,i} \otimes x_{2,i} \in U_1 \otimes U_2$. In Sweedler's notation this is written as $(x_{(1)}, x_{(2)})$ with the summation, indices, and tensor symbols all dropped. Provided we make sure that any expression in which (x, y) appears is linear in both variables, this seeming recklessness introduces no errors. This practice is what Shulman develops into the core of his practical type theory.

To see how this plays out, suppose that we have a morphism f with domain A and codomain consisting of the tensor product $(B_1, ..., B_n)$. In Shulman's adaptation of Sweedler's notation this corresponds to the judgement:

$$a : A \vdash \left(f_{(1)}\left(a\right), \ldots, f_{(n)}\left(a\right)\right) : (B_1, \ldots, B_n)$$

Now, as our rules which derive typing judgements come from the generating arrows in the set X_1 unless we find in X_1 some $p : (B_1, \ldots, B_n) \to B_k$ which acts like projection, a property encoded by relations in R, we simply cannot derive the judgment $a : A \dashv f_{(k)} : B_k$. Unless our presentation provides us with projection maps, we do not have access to them.

Since this type theory allows us to pretend, in a sense, that types were "sets with elements", the symmetric monoidal aspect of the type theory is that:

- terms of product types are not-necessarly-decomposeable tuples;

whereas in a cartesian "sets with elements" style type theory:

- terms of product types are decomposeable tuples.

Remark. The purpose of the "scare parentheses" around the sub-scripts is to remind us that we cannot treat the $x_{(k)}$ individually - a priori $(x_{(1)}, \ldots, x_{(n)})$ is not decomposeable into a list.

5 Example: The Free Dual Pair

5.1 A Presentation for the free dual pair

Recall that for a vector space V and its dual vector space V^*, we've a bijection

$$\mathsf{Hom}\left(A \otimes V, B\right) \xrightarrow{\sim} \mathsf{Hom}\left(A, V^* \otimes B\right)$$

natural in vector spaces A and B. This example is abstracted into the usual defintion of a dual pair in a symmetric strict monoidal category as follows.

Definition 1. A dual pair $(D, D^*, \mathsf{coev}, \mathsf{ev})$ in a symmetric monoidal category $(\mathcal{C}, (\_, \_), ())$ is comprised of:

- a pair of objects D and D^* of $\mathcal{C}$;

- a morphism $\mathsf{coev} : \mathbf{1} \longrightarrow D \otimes D^*$; and

- a morphism $\mathsf{ev} : D^* \otimes D \longrightarrow \mathbf{1}$

satisfying the triangle identities:

$$\begin{tikzcd}
D \arrow[dd, equal] \arrow[dr, "(\mathsf{coev}, D)"] & & D^* \arrow[dd, equal] \arrow[dr, "(D^*, \mathsf{coev})"] & \\
& (D, D^*, D) \arrow[dl, "(D, \mathsf{ev})"] & & (D^*, D, D^*) \arrow[dl, "(\mathsf{ev}, D^*)"] \\
D & & D^* &
\end{tikzcd}$$

These these data suggest a presentation $(\mathsf{R}, \mathsf{X}_1, \mathsf{X}_0)$ for the PROP generated by a dual pair. We set:

- $\mathsf{X}_0 = \{ D, D^\star \}$;
- $\mathsf{X}_1 = \{ \mathsf{coev} : () \longrightarrow (D, D^*) , \mathsf{ev} : (D^\star, D) \longrightarrow () \}$; and
- $\mathsf{R} = \{ \mathsf{triangle} : \mathrm{id}_D = (D, \mathsf{ev}) \circ (\mathsf{coev}, D), \mathsf{triangle}^* : \mathrm{id}_{D^\star} = (\mathsf{ev}, D^\star) \circ (D^\star, \mathsf{coev}) \}$

5.2 The Type theory of the free dual pair

While we haven't bothered with them until now, practical type theory has rules for term formation. In the case of the practical type theory for the free dual pair, the rules for the term judgment are few. We present a streamlined version of them:

- $\dfrac{(x : A) \in \Gamma}{\Gamma \vdash x\ \mathsf{term}}$ which is the usual variable rule;

- $\dfrac{1 \leq k \leq 2}{\mathsf{coev}_{(k)}\ \mathsf{term}}$ which allows us to form the terms with which we'll be able to derive co-evaluation as a typing judgement; and

- $\dfrac{\Gamma \vdash M\ \mathsf{term} \quad \Gamma \vdash N\ \mathsf{term}}{\Gamma \vdash \mathsf{ev}\,(M, N)\ \ \mathsf{term}}$ which allows us to form the terms with which we'll be able to derive evaluation as a typing judgement.

FILL IN ADMISSIBLE RULES FOR TYPING JUDGEMENTS

note that we may derive the typing judgments corresponding the compositions

$$(D) \xrightarrow{(\mathsf{coev}, D)} (D, D^*, D) \xrightarrow{(D, \mathsf{ev})} (D)$$

and

$$(D^*) \xrightarrow{(D^*, \mathsf{coev})} (D, D^*, D) \xrightarrow{(D, \mathsf{ev})} (D)$$

Using the admissible rulessing the actual rules for the type theory (modulo the consideration of 'activeness') the canonical derivation for that first morphism follows.

$$\dfrac{\dfrac{\mathsf{coev} \in \mathcal{G}\left(; D, D^*\right) \quad \mathfrak{a} \in \mathfrak{A} \quad (132) : (D, D, D^*) \xrightarrow{\sim} (D, D^*, D)}{x : D \vdash \left(\mathsf{coev}^{\mathfrak{a}}_{(1)}, \mathsf{coev}^{\mathfrak{a}}_{(2)}, x\right) : D} \qquad \mathsf{ev} \in \mathcal{G}(\left(D^*, D\right), ())}{x : D \vdash \left(\mathsf{coev}^{\mathfrak{a}}_{(1)} | \mathsf{ev}\left(\mathsf{coev}^{\mathfrak{a}}_{(2)}, x\right)\right)}$$

Lastly, we have the axioms

$$\dfrac{\mathsf{triangle} \in \mathsf{R}(thethings)}{M : D \vdash \left(\mathsf{coev}_{(1)} | \mathsf{ev}\left(\mathsf{coev}_{(2)}, M\right)\right) = M : D}$$

and

$$\dfrac{\mathsf{triangle}^{\star} \in \mathsf{R}(thethings)}{N : D^* \vdash \left(\mathsf{coev}_{(2)} | \mathsf{ev}\left(N, \mathsf{coev}_{(1)}\right)\right) = N : D^*}$$

rules enough to generate a congruence.

5.3 'Elements' of dual objects

We have now developed a type theory for the free dual pair which endows the dual objects D and D^* with a universal notion of element (D-typed term and $D^{\star}$-typed term respectively). Since the notion of dual pair abstracted the instance of a pair of dual vector spaces, which in particular have actual elements, it behooves us to ask:

“how much like a vector is a term of type D?”

The answer is both practical and electrifying (though perhaps the authors of this blog post are too easily electrified).

It’s easy enough to believe that the evaluation map

$$\mathsf{ev} : (D, D^{\star}) \longrightarrow ()$$

endows the terms of type D, or D^* for that matter, with structure of scalar valued functions on the other. The triangle idenities impose the unique determination of terms of type D or D^* in terms of their values as given by ev.

Consider that, for a finite dimensional vector space V over a field k, a basis $\{\mathbf{e}_i\}_{i=1}^n$ for V and a dual basis $\{\mathbf{e}_i^*\}_{i=1}^n$ for V^* give us an elegant way to write coev and the the first triangle identity. We write

$$\begin{tikzcd}
k \arrow[r, "\mathsf{coev}"] & V \otimes V^* \\
x \arrow[r, mapsto] & \sum_{i=1}^n \mathbf{e}_i \otimes \mathbf{e}_i^*
\end{tikzcd}$$

and see that

$$\begin{tikzcd}
V \arrow[dd, equal] \arrow[dr, "\mathsf{coev}\otimes V"] & \\
& V \otimes V^* \otimes V \arrow[dl, "V\otimes\mathsf{ev}"] \\
V &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbf{v} \arrow[dd, mapsto] \arrow[dr, mapsto] & \\
& \left(\sum_{i=1}^n \mathbf{e}_i \otimes \mathbf{e}_i^*\right) \otimes \mathbf{v} \arrow[dl, mapsto] \\
\mathbf{v} = \sum_{i=1}^n \mathbf{e}_i^*\left(\mathbf{v}\right) \cdot \mathbf{e}_i &
\end{tikzcd}$$

The observation, for dual vector spaces defined by way of the equality

$$V^* = \mathsf{Hom}_{\mathsf{Vect}_k}(V, k)$$

that the triangle identities hold is just the observation that a vector is precisely determined by its values: every vector $\mathbf{v}$ is equal to the un-named vector

$$\sum_{i=1}^{n} \mathbf{e}_i^* (\mathbf{v}) \cdot \mathbf{e}_i$$

which is defined by taking the values $\mathbf{e}_i^* (\mathbf{v})$ at the dual vectors $\mathbf{e}_i^*$. As part of the definition of a dual pair in an arbitrary symmetric strict monoidal category then, the triangle identities imposes this as a relationship between ev and coev. But within type theory, this sort of relationship between an un-named function and its values is familiar, indeed it is something very much like β-reduction.

To see this more clearly, let's make a pair of notational changes to bring the parallel to the fore. In place of writing

$$(x, y) : (D^*, D) \vdash \mathsf{ev}(x, y) : ()$$

we'll denote ev infix by $\_ \triangleleft \_$ and write

$$(x, y) : (D^*, D) \vdash x \triangleleft y : () \, .$$

Similarly, in place of writing

$$\vdash \left(\mathsf{coev}_{(1)}, \mathsf{coev}_{(2)}\right) : (D, D^*)$$

we'll denote coev by the pair $\left(u, \lambda^D u\right)$ and write

$$\vdash \left(u, \lambda^D u\right) : (D, D^*) .$$

With this choice of notation then, the axiom which corresponds to the first triangle identity is

$$x : D \vdash \left(u | \lambda^D u \triangleleft x\right) = x : D.$$

Then, as Shulman points out, since $=$ is a congruence with respect to substitution, if we've, for some term M, the term $\lambda^D u \triangleleft M$ appearing in the scalars of a list of terms, then we may replace all instances of u in the rest of the term with M. While a mouthful, this is a sort of 'β-reduction for duality' as Shulman identifies it. Conceptually interesting in its own right, this obeservation also yields a one line proof for a familiar theorem.

Lemma 2. (cite original result)(cyclicity of trace)Let $(\mathsf{C}, (\_, \_), ())$ be a symmetric strict monoidal category, let

$$\left(A, A^*, \left(u, \lambda^A u\right), \_ \triangleleft \_\right)$$

and

$$\left(B, B^*, \left(v, \lambda^B v\right), \_ \triangleleft \_\right)$$

be dual pairs in C, and let $f : A \longrightarrow B$ and $g : B \longrightarrow A$ be morphisms in C. Let $\mathsf{tr}\,(f \circ g)$ be the composition

$$() \xrightarrow{\left(v, \lambda^B v\right)} (B, B^*) \xrightarrow{f \circ g} (B, B^*) \xrightarrow{(12)} (B^*, B) \xrightarrow{\_ \triangleleft \_} ()$$

and likewise let $\mathsf{tr}\,(g \circ f)$ be the composition.

$$() \xrightarrow{\left(u, \lambda^A u\right)} (A, A^*) \xrightarrow{g \circ f} (A, A^*) \xrightarrow{(12)} (A^*, A) \xrightarrow{\_ \triangleleft \_} () .$$

Then, $\mathsf{tr}\,(f \circ g) = \mathsf{tr}\,(g \circ f)$.

Proof.

$$\begin{aligned}
\mathsf{tr}\,(f \circ g) &\overset{\mathsf{def}}{=} \left(\middle|\lambda_u^B \lhd f\,(g\,(u))\right) \\
&= \left(\middle|\lambda_u^B \lhd f\,(v)\,, \lambda_v^A \lhd g\,(u)\right) \\
&= \left(\middle|\lambda_v^A \lhd g\,(f\,(v))\right) \\
&\overset{\mathsf{def}}{=} \mathsf{tr}\,(g \circ f)
\end{aligned}$$

Where the judged equalities are application of 'β-reduction for a duality'. □