0.1. **From domain to foundations.** While in the cases of HoTT or intuitionistic linear logic we have a particular form of reason or computation we wish to model, and only afterwards are we trying to use these type theories to reason *about* $(\infty, 1)$-toposes or closed symmetric monoidal categories, Shulman's project with practical type theory is to start with a domain and reverse engineer a syntax; we have a sort of category, symmetric monoidal categories, and wish to ask them to provide us with a type theory. This is not to say however that we have *no* hand in specifying the type theory.

0.2. **Criteria for practicality.**

- syntax semantics correspondence satisfying the following criteria...

First, for any symmetric monoidal category, not merely concrete and familiar ones, the type theory we get should permit us to leverage our intuition and experience with reasoning about sets with elements[1]. As such our type theory and its semantics should correlate

$$\begin{array}{rcl} \text{objects} & \longleftrightarrow & \text{contexts} \\ \text{morphisms} & \longleftrightarrow & \text{typing judgements} \end{array}$$

Second, we must not:

> "(strike) a Map of the Empire whose size was that of the Empire"
> - J.L.Borges "On Exactitude in Science"[2]

so we ask that this type theory be specified not by a symmetric monoidal categories directly but instead by elegant presentations thereof.

Lastly, we wish to be assured that the theory is complete in the sense that everything which holds in a particular symmetric monoidal category is derivable as a judgement.

0.3. **Sweedler's notation for co-algebras and Shulman's term judgements.** We cannot simply bend some cartesian type theory to our will to develop a practical type theory for symmetric monoidal categories, because, while $\times$ enjoys a universal property characterized in terms of projection maps, the universal property of $\otimes$ is characterized in terms of co-projection maps. So, while in a cartesian monoidal category like Set, for any $z \in X \times Y$, we have that[3]

$$z = \mathsf{pair}\left(\mathsf{pr}_1\left(z\right), \mathsf{pr}_2\left(z\right)\right)$$

no such characterization of the elements of $X \otimes Y$ is immediately available.

Locating the concern in the familiar terrain of vector spaces provides this categorical phenomenon with concrete intuition: the tensor product $\bigotimes_{i\in\{1,\ldots,n\}} V_i$ of vector spaces is comprised not of simple tensors

$$v = v_1 \otimes \cdots \otimes v_n = \mathsf{tuple}\left(v_1, \ldots, v_n\right)$$

[1] A possibly enlightening parallel to this program can be found in the history of algebraic geometry. Just as the Zariski topology extends the interpretation, beyond the rings of polynomial functions on algebraic varieties, of rings as rings of functions on spaces, a philosophical perspective on the goal of Shulman's practical type theory extends the formal metaphor of symmetric monoidal categories of vector spaces to arbitary symetric monoidal categories.

[2] https://kwarc.info/teaching/TDM/Borges.pdf

[3] Type theoretically, this is precisely the derived judgement of the η-computation rule for the product type.

but of linear combinations of them. We can however pretend that every tensor is a tuple, provided we are very careful; this is exactly the power of Sweedler's notation for co-algebras which Shulman's expands into a syntax for symmetric monoidal categories.

The co-multiplication of a co-algebra in vector spaces may be written elementarily as follows.

$$\begin{array}{ccc} C & \xrightarrow{\quad\triangle\quad} & C \otimes C \\ c & \longmapsto & \sum_{i=1}^{k} c^i_{(1)} \otimes c^i_{(2)} \end{array}$$

Sweedler long ago noted that it was convenient to drop summation and even to drop indices. Going further and replacing $(\_) \otimes \cdots \otimes (\_)$ with $(\_, \ldots, \_)$, both in the formation of tensor products and for 'simple tensors', we then denote maps into arbitrary tensor products as follows.

$$\begin{array}{ccc} A & \xrightarrow{\quad f\quad} & (B_1, \ldots, B_n) \\ a & \longmapsto & \left(f_{(1)}(a), \ldots, f_{(n)}(a)\right) \end{array}$$

Shulman translates this notation into term formation rules: for example[4]

$$\frac{\Gamma \vdash M_1 \ \mathsf{term} \ldots \Gamma \vdash M_m \ \mathsf{term} \qquad f \in \mathsf{Hom}(A_1, \ldots, A_m; B_1, \ldots, B_n) \ \ m \geq 1 \ n \geq 2 \ 1 \leq k \leq n}{\Gamma \vdash f_{(k)}(M_1, \ldots, M_m)}$$

which in the case $m = 1$ and $\Gamma \vdash a \ \mathsf{term}$ allows us to form the bare terms $f_{(k)}(a)$ as in Sweedler's notation and then the rules for our typing judgements, which we'll cover later, will allow us to derive

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

the typing judgement which corresponds to the morphism f.

What's more, just as the syntactic limits on defineable expressions in HoTT grant that any expression is born invariant, the term formation rules of our type theory will make sure that only permissable expressions using Sweedler's notation will be syntactically accessible.

0.4. **A suitable choice of presentation.** It is 'well known' that every symmetric monoidal category is equivalent to a symmetric strict monoidal category. Signifigantly less well known is that every symmetric strict monoidal category is equivalent to a PROP.

Definition 1. A **PROP** $(P, \mathscr{P})$ is comprised of:

- a set P of generating objects; and
- a symmetric strict monoidal category $\mathscr{P} = (\mathscr{P}, \otimes, \mathbf{1})$ for which $(\mathsf{Ob}(\mathscr{P}), \otimes, \mathbf{1})$ is the free commutative monoid on the set P.

A morphism of props is a pair $(f, F) : (P, \mathscr{P}) \longrightarrow (Q, \mathscr{Q})$ where $f : P \longrightarrow Q$ is a function and $F : \mathscr{P} \longrightarrow \mathscr{Q}$ is a strong symmetric monoidal functor with $\mathsf{Ob}(F)$ being the morphism of commutative monoids induced by the function f. We denote the category of PROPs by PROP.

[4]For expository purposes we've replaced $f \in \mathcal{G}(A_1, \ldots, A_m; B_1, \ldots, B_m)$, which is the hypothesis that f is a *generating* morphism, with the more general hypothesis that $f \in \mathsf{Hom}(A_1, \ldots, A_m; B_1, \ldots, B_n)$.

Indeed we've an equivalence of categories

$$\mathsf{SymStrMonCat}_{\mathsf{strong}} \xrightarrow{\sim} \mathsf{PROP}$$

The utility of this observation for our ends is two-fold:

- syntactically, finite lists up to symmetry are far si
- a presentation of a PROP need only be comprised of three sorts of data:
  - a generating set of objects;
  - a generating set of morphisms, formal arrows between finite lists of generating objects;
  - a generating sets of equalities of morphisms, between formal composites and tensors generating morphism.

An elegant packaging of this second fact follows.

Definition 2. Let a **signature** for a prop $\mathcal{G} = (G_0, G_1)$ be comprised of a pair of sets:

- G_0 a generating set of objects; and
- G_1 a set of formal arrows $f : (A_1, \ldots, A_n) \longrightarrow (B_1, \ldots, B_m)$ where the A_i and B_j are of G_0.

Remark 3. AThose familiar with the theory of computads will note that the data enumerated above are exactly a 2-computad for a PROPs.

0.5. **On initiality and the admissability of structural rules.** In this framework the completeness result desired takes the form of an *initiality theorem*, a theorem which here will posit that:

- for $\mathscr{C}$ be a symmetric monoidal category and $\mathbf{T}_{\mathscr{C}}$ the associated type theory,
- the term model of $\mathbf{T}_{\mathscr{C}}$, the initial semantic interpretation of $\mathbf{T}_{\mathscr{C}}$, is equivalent as a symmetric monoidal category to $\mathscr{C}$.

In particular this

1. Architecture of the paper

1.1. **Idea of the content of the paper.** What is the general idea for obtaining this practical type theory?

(1) *Start with some input data (which we will call signature)*
   - Our input data for constructing type theories will consist of a set of objects together with a set of arrows whose domain and codomain consist of finite lists of objets.

(2) *We build a type theory for the free prop generated by a signature*
   - This is done by defining rules for terms and rules for typing judgements.
   - Much care is taken while defining these rules to ensure, among other things, that the composition and the exchange rules are admissible, therefore ensuring that any judgement has a unique derivation. This is a key requirement to prove the initiality theorem that we discussed earlier.

(3) *The term model of this type theory can be proven to be the prop freely generated by the input signature. Hence we now have the initiality theorem*

- Taking the contexts of this type theory as objects and the derivable term judgement (modulo an equality rule) as morphisms forms a strict symmetric monoidal category, which we denote $\mathfrak{FG}$. It is also easy to show that $\mathfrak{FG}$ is in fact also a prop.
- We can then show that $\mathfrak{FG}$ is actually the free prop generate by $\mathcal{G}$.

Great. Now we have a type theory for props freely generated by a signature. But, what about all the other props? Or in other words, how do we deal with SMC that have additional equality relations? Well, since we know that the category of prop is monadic over the category of signatures, we have that every prop $\mathcal{P}$ admit a presentation in terms of signatures (i.e. a coequaliser diagram $\mathfrak{FRFG} \to \mathcal{P}$), where $\mathcal{R}$ is a signature that provides equality axioms.

(4) *We then essentially redo the two previous steps, but this time we also quotient by these additional equality axioms provided by the signature $\mathcal{R}$ in the presentation of the prop*

That's it! Following these steps gives us a type theory for the prop presented by $(\mathcal{G}, \mathcal{R})$, which, as proven in the paper, allows us to reason about structures in any props, and hence also in any symmetric monoidal category.

1.2. **Illustrating the content of the paper using the free dual pair example.** Start with the duality $(D, D^*, \mathsf{ev}, \mathsf{coev})$ (and its associated free dual pair $\mathcal{D}$) given above.

(1) *Start with some input data*
- Starting with the duality $(D, D^*, \mathsf{ev}, \mathsf{coev})$, we define some input data (signature) $\mathcal{G}$ as follows:
  - Objects: $\{D, D^*\}$
  - Arrows: $\{ev : (D^*, D) \to (), \quad coev : () \to (D, D^*)\}$

(2) *Build a type theory from the input data*
- The arrows in the input data become the rules in the type theory $\mathbf{T}_\mathcal{G}$:

$$\mathsf{ev} : (D^\star, \cancel{D}) \cancel{\longrightarrow} () \dashv \qquad\qquad \mathsf{coev} : () \cancel{\dashv\vdash} (\cancel{D^\star, D}) \dashv$$

***HOW CAN WE EXPRESS THE EQUALITY RULE IN THIS EXAMPLE?

(3) *Prove initiality theorem*
- By the theorems in the paper the context and derivable typing judgements in the type theory $\mathbf{T}_\mathcal{G}$ form a prop which is in fact the free prop generated by the input data $\mathcal{G}$.

(4) *Account for equality relations by using a presentation as a coequaliser*
- The free dual pair $\mathcal{D}$ admits a presentation as the following colimit:

$$\varinjlim \{\mathfrak{FRFG}\} \xrightarrow{\sim} \mathcal{D}$$

where R is the signature of relations that imposes the following two axioms (these correspond to the commutative diagrams introduced above):

$$(\mathsf{ev} \otimes \mathsf{id}_D) \circ (\mathsf{id}_D \otimes \mathsf{coev}) = \mathsf{id}_D \qquad\qquad (\mathsf{ev} \otimes \mathsf{id}_D) \circ (\mathsf{id}_D \otimes \mathsf{coev}) = \mathsf{id}_D$$

First translation:

$$x : D \vdash (\eta_{(1)} \mid \varepsilon(\eta_{(2)}, x)) = x : D \qquad y : D^* \vdash (\eta_{(2)} \mid \varepsilon(y, \eta_{(1)})) = y : D^*$$

Second translation:

$x : D \vdash (u \mid \lambda^D u \triangleleft x) = x : D \qquad y : D^* \vdash (\lambda^D u \mid y \triangleleft x) = D^*$

***NEEDS FIXING!!!