MS1GP Presentation Speaker Notes *March 11, 2023*

Matthew Drury

- TODO

MS1GP Presentation Speaker Notes *March 11, 2023*

Ben Brook

- TODO

MS1GP Presentation Speaker Notes *March 11, 2023*

Oliver Dixon

- To start, we need to define some more theory (not much!). Follow the slide prompts for construction of the general monoidal category with a base category of $\mathcal{C}_0$.
- We need these natural transformations to induce certain properties on the bifunctor $\otimes$, and it may not possess these natively. In other words, these diagrams must commute [show the CDs very briefly].
- We sometimes omit the natural transformations from the tuple, as their particulars are seldom of especial importance.
- Monoids are objects in a monoidal categories together with two objects. [Emphasise the importance of μ having a domain of $M \times M$, as it is integral for the monad application is Haskell.]
- Again, these diagrams must commute [show very briefly].
- There's not much to say about functional programming in its most general sense besides the stipulation that functions *cannot induce side-effects*; they must only work with the data they are given, and return an explicit transformation of that data. We're beginning to understand why strong type systems, as those in functional languages, interface so nicely with Category Theory!
- We have already seen endofunctors in MD's section, so considering the category of such objects should not be difficult. We've also seen natural transformations, although bifunctors have not yet been (explicitly) mentioned.
- Introduce Haskell as the "canonical" functional language, whose purity facilitates these direct connexions with Category Theory. Consider $\mathbf{Endo}\,(\mathbf{Hask})$, which is semantically consistent, since monoidal categories (such as **Hask**) require a base category, the product of which its bifunctor can use as its domain.
- Reiterate the notion of a shared mutable state, and explain how weaving through imperative-like sequential call structures is possible with monads, due to the category product being used to encode the function parameter and current context.

  $$\otimes\colon\ \mathbf{Hask} \times \mathbf{Hask} \to \mathbf{Hask}$$

  This is the core idea of enabling stateful computation. We will not have time to discuss *the* λ-calculus in detail, unfortunately, but its theory (developed by Alonzo Church, Alan Turing's supervisor at Cambridge!) does form the basis of much of functional paradigms, even in their modern forms.
- Take questions.