Applied Compositional Thinking for Engineers (ACT4E)

Guest Lecture 2 - Dr. Brendan Fong

Questions & Answers

Q: What is the advantage of thinking in 2-morphisms vs thinking in Functors?

JL: Functors are most immediately thought of as the “morphisms between categories”. There is also a analogous notion of morphism between 2-categories (a 2-functor!). 2-morphisms, on the other hand, are “morphisms between morphisms” *inside* a 2-category. (See also the question below.)

All of these definitions though can manifest differently in different contexts!
As a simple example of this phenomenon that we saw already in the course: a set can be an object (in the category of sets, for example), but in a different “context”, a set can also be a morphism (e.g in the category of relations).

Q: I’m not familiar with 2-categories, but from the notation: what is the relationship with natural transformations? Those morphisms of morphisms seem so similar.

GZ: There are various examples in which 2-morphisms are natural transformations:
E.g. in the category of categories (also called Cat), objects are categories, morphisms are functors, and 2-morphisms are natural transformations.
JL: side note: we will introduce functors and natural transformations in the ACT4E course soon!
JL: For comparison, here is also an example of a 2-category that does not have natural transformations as its 2-morphisms. There is a 2-category where the “0-dimensional objects” are sets, the 1-dimensional morphisms are relations, and the 2-morphisms are inclusions between relations (as we saw is possible in the lecture). There is also variant of this construction where 0-dimensional objects are vectors spaces, 1-morphisms are linear relations, and 2-morphisms are again inclusions.

Q: monoidal categories assume a flow from left to right or they can model feedback?

GZ: Good question, which we will largely answer throughout the course. To model feedback, we will use what's called a *traced monoidal category*, which on top of being monoidal, has the properties of being *symmetric* and being *traced*. We will have dedicated lectures for this.

MH: Great, thx. Will we also discuss modelling feedforward paths?

Q [zoom, KK] Can functors capture emergent semantic effects of composite wholes that might arise from a collection of part semantics?
Possible alternative question: Can functors capture non-obvious emergent features(emergent semantic effects) of a system composed of many elementary parts (part semantics)?
AR: Could you clarify what you mean by "emergent semantic effects"?
KK: I think what I mean is, if a composite whole in a hierarchical construction had properties greater than the sum of its constituent parts, is there a particular functor or structure that you could communicate in that process, from collection of parts to new whole with different properties. E.g. a collection of atoms in a pure metal, like Cu, that come together in a particular pattern (fcc crystal) to form grains, which form a further grain structure.
AR: Could you also give an example of interest to you? (I find examples really help my understanding)
NM: Are you talking about strong or weak emergence? (e.g. are you interested in things that can or cannot be predicted from computations of the fine scale behaviour, e.g. atoms)
KK: Both are interesting! But I think perhaps weak emergence, based on your definition.
NM: Does the alternative question sound like what you are interested in?
KK: Yes. Thanks for asking it better! :)

I'm not the one who asked but I can clarify:

"In philosophy, systems theory, science, and art, emergence occurs when an entity is observed to have properties its parts do not have on their own, properties or behaviors which emerge only when the parts interact in a wider whole.

Emergence plays a central role in theories of integrative levels and of complex systems. For instance, the phenomenon of *life* as studied in biology is an emergent property of chemistry, and psychological phenomena emerge from the neurobiological phenomena of living things."

https://en.wikipedia.org/wiki/Emergence

Q: How would the model of pandemics in Catlab differ from a model in Matlab/Simulink or similar?
Q (followup to above): The model representation can be programatically manipulated. (The model becomes data). What is the "category" that describes the model manipulation language?

Q: How do we, as “Compositional Thinkers” handle the situation where the interpretation/abstraction breaks down or leaks? For example, I’ve heard a story of a GA designed radio which had no explicit or apparent antena but *worked*; this was later determined to be due to the natural antena formed by the ground (or something like that).

Q: What does it mean to be “complete” regarding the tools or category theory identifying those equivalencies?

JL: In what context/slide/topic was the word “complete” mentioned, or how is this word meant here?
AR: “sound and complete reasoning about matrices”

Q: You say it’s useful as a new proof tool; is it also useful as a computation tool to “typecheck” your computation? E.g. as in a suped up version of [Bra-Ket Notation Trivializes Matrix Multiplication](#)?

Q: Is there a "graphical" method to construct the inverse of a signal flow graph that corresponds to the inverse of its matrix (if it exists)
AR: From reading https://graphicallinearalgebra.net/ (which is a slow and gentle intro into this category), I’m pretty sure there is a graphical method to *represent* an inverse matrix--it’s the matrix that multiplies to 1. There seems to be a way to compute it, but I’m less sure about that.

Q: If we can think of functors as giving semantics to syntax, then what intuition can we use to think of profunctors?

Q: Are there general “compilers” that would simplify signal networks (e.g. minimise circuit hardware size, optimise machine code, simplify logical/scientific relations)? And are there ways to control and allow for different optimisation targets?
OP: this may be more general than compilers

Q [from zoom SV] Can we describe a monoidal category in terms of operads ?

NM: yes, an operad generalises monoidal categories since there exists an object that is the monoidal product of every (ordered) pair of objects, but operads don’t have to have a monoidal product. However, if constraints are put on the sorts of an operad to recapture the monoidal product structure then you can represent any monoidal category as an operad (but not vice-versa)

Q [from zoon FE] Looking from a computational engineering perspective i would expect a little bit more expression power from a new math theory than just structure preserving mappings and linear mappings. This is for me the trivial case. I am not sure if I have a mistake in my understanding. I simply make the shortcut analogy to say that pseudocoding is "the" advanced math you might be looking for.

NM: maybe limits might be what you are interested in