Applied Compositional Thinking for Engineers (ACT4E)

Session 11 - Feedback

Questions & Answers

Q. When does the strictification theorem apply, and what more precisely does it say? For example, in programming languages, you often care about the distinction between (x, y) and (y, x), even though they're isomorphic.
This might be good question to answer here as well

NM: The definition can be found here:
https://ncatlab.org/nlab/show/coherence+theorem+for+monoidal+categories
Regarding the intuition I don't feel I have a deep intuition so I can't really say anything (I'd think about it as quotienting by the isomorphisms, but the link I've just given says this approach is wrong and leads to confusion).
In particular definition 4 is what it says: Every monoidal category is monoidally equivalent to a strict monoidal category. And so holds for every monoidal category

AR: What does it mean in this context to be "monoidally equivalent"?

NM: I think they mean there is an equivalence of categories (https://ncatlab.org/nlab/show/equivalence+of+categories) which respects the monoidal structure. That is to say there exists a functor F from every monoidal category A to a strict monoidal category B where the monoidal structure of A is mapped to the monoidal structure of B. There is also a functor G in the opposite direction that you can kind of think of as the inverse of F that also respects the monoidal structure. However the technical definition is that FG and GF have natural isomorphisms to the identity functors on categories A and B.
AR: I see, so basically, these functors are not guaranteed to preserve other properties, which may themselves be of interest, like the value to which a program evaluates?

NM: This is getting beyond where I feel comfortable making any claims, but I don't think so, (it will depend on how you are defining the values, if they are elements of an object or object itself) the pair of functors will map back onto the same objects but I feel this is better left as a question for the end of the talk.

JL: here is also a concise discussion:
https://unapologetic.wordpress.com/2007/07/01/the-strictification-theorem/

Q: Do evaluation and coevaluation have some relationship to unit and counit in adjunction?

NM: yes, but I'm not sure exactly how to define it, I suspect you'd make use of the fact that a monoidal category is a 2-category with one object, then the objects of the monoidal category happen to be (endo)functors in the 2-category. This diagram might be more helpful to see a connection with the string diagrams: https://en.wikipedia.org/wiki/File:String_diagram_adjunction.svg you read left to right and the edges represent the functors between categories.

LM: I was thinking about evaluation and exponentials being related to adjuncion

JL: Yes, there is a notion of adjunction inside of a bicategory, and this generalizes both the notion of adjunction that we saw for functors, and the notion of duals/evaluation/coevaluation for (symmetric) monoidal categories.

Q: Would a real number then be represented as a box with no wires?

NM: If you are thinking about vector spaces as your objects then you'd see it as a box with no wires, however you can think about this as a box with a wire in and out that is labelled by the unit object (which is R or C for vector spaces), I've seen this sometimes written with dotted lines to indicate that you can either keep or forget them (though I can't remember if this is more tensor network notation). The evaluation and co-evaluation can also be seen as having these dotted lines on the side that has nothing as well but you can just forget about it without losing anything.

Q: Can these string diagrams be practically used to aid correct linear algebra calculation on paper? A la Bra-Ket Notation Trivializes Matrix Multiplication?

GZ: See example in lecture given by Jonathan

Q: Is there any study of "diagrams with holes", which I assume would represent functors?

AR: "templates" sounds right. I'm thinking like "contexts" in PL theory.

Q: [slide 29] "How can we compute it's trace?" The diagram on the slide doesn't seem to *compute* anything, as much as *describe* it. What am I missing?

NM: you can read the diagram from left to right: as R -> V \otimes V* -> V \otimes V* -> R
The first arrow is preparing the vector \sum_{i} v_{i}\otime v_{i}^* where v_{i} is a basis for vector space V.
The second arrow is the matrix multiplication so you now have \sum_{i} f(v_{i})\otime v_{i}^*
Finally the last arrow is an inner product with \sum_{i} v_{i}^*\otime v_{i} which gives back \sum_{i} f_{i,i}

AR: That is, this diagram proves that a specific map is the trace; I don’t see how it shows you that, e.g. Tr(I_2)=2