Applied Compositional Thinking for Engineers (ACT4E)

Guest Lecture 4 - Prof. Michael Johnson

Questions & Answers

Q: Would what you say have held if the class was about Compositional Thinking rather than Computational Thinking?
MJ: Nothing changes, switch words :) *Sorry about that! I really meant* compositional *thinking.*

Q: could you be more precise in what you are counting as being in the first couple of lectures?
GZ: This lecture is self-contained
OP: I was thinking for people who are familiar with category theory but to allow us to know what to focus on when trying to apply these ideas
Let me reconsider this as "what are the basic ideas of category theory we use in applications?"
In my experience: the definition of category, and especially commuting diagrams and the way category theorists think using them; finite limits; monos and epis; functors; natural transformations. That's about it. Very basic ideas, but very useful. We also use (quite a lot) fibrations and opfibrations, finite coproducts, and more recently lenses (functors and cofunctors that 'match' on objects).

Q: How do you deal with "almost" commuting diagrams? The example I have is the Set type in Haskell, which doesn't quite represent a mathematical set but a bunch of objects de-duplicated by some rule; this means that mapping a composite function over the set is not necessarily the same as mapping the two parts of the composite in sequence (specifically, if intermediate set would be de-duplicated but the final set would not be).

Wow, this is such an interesting question I've taken a long time to answer it. The reason it interests me so much is that it is related to one of the topics I mentioned at the very end --- lenses. The lenses I work with are usually what Hoffman, Pierce and Wager call "very well behaved" which means that they satisfy the so-called Put-Put law. In the situation you describe, as long as the last thing I did was de-duplicate (even if I had already de-duplicated between the two mappings), I would get the same answer wouldn't I? That would be an example of satisfying the Put-Put law, but I haven't expressed it yet in a way that you might find very relevant to your situation. Still, I'm very glad you asked!

Q: If I have an entity-relation diagram seen as a category, can we see the subset of the "commuting triangles" for expressing rules as a subcategory of the category?
I think I answered a slightly different question about subcategories more generally. Still, I agree that we can (although for communication purposes, especially with applications people, it's often better to use the 'natural' commuting diagram (which might be a quadrilateral or pentagon or…) -- every such diagram can be captured with triangles, but it's often easier to understand in one larger chunk).

Q: If you have a network of constraints/relations and you change one of the constraints, is it possible to propagate the effects on the network? Is it possible to find a way in which other nodes should adapt in order to "maintain" the relations?
This is a particularly tricky, but important issue, that I talked about a bit. There are certainly some cases where we can deal with things automatically, but we don't have the general theory that I think would be useful.

Q: What do you think are promising subfields of applied category theory which an engineer should learn/focus on?
A very biased answer: Model Driven Engineering, fibrations, lenses, spans and cospans. But always keep in mind that the simple "category theoretic way of thinking" takes you a long way.

Q: what are the funny arrow shapes in the Telstra diagram?
I apologise for those -- they were taken from the original Telstra report. Telstra wanted cardinalities *explicitly shown (so the extra lines on the arrows were really just indicating how many instances were involved at each end of the arrow). Thanks for asking, and sorry I didn't explain in the limited time available.*

Q: can all these constraints be expressed in SQL (field REFERENCES table(field)) or is it more advanced semantics?
The relationship with SQL is quite interesting, especially for those learning category theory. Structural SQL queries (those that don't involve a calculation like count() *or* average() *) correspond to limits in the abstract category that is the database spec. (I hope that makes sense)*

Q: Are there any references for the category theory point of view of constraints in databases? I know of D Spivaks work on categorical databases, but specifically for constraints what are good resources to read more?
There are some in my publication list at **www.cs.mq.edu.au/~mike**

Q: I believe you were saying that the arrow between the pullback and the product in logical constraint was a monomorphism. Was that using the universal property of the product?
Just to clarify (sorry if I didn't express it clearly): The arrow comes from the universal property of the product. It is the mediating morphism. The extra requirement for the constraint is that that arrow, which came automatically from the product, has to in fact be a monomorphism (which is not in general necessarily true).

Q: Are your categorical models of databases (with commutative constraints, and w/o the actual data) the same thing as David Spivak's OLOGs?
Very nearly the same. Certainly the same underlying ideas. There is a technical difference to do with coproducts that I think is important for my work, but that David seems quite happy without.

Q: what are examples of Functors usage in consultancy work that were talked about in the lecture? Just curious
The most important use of functors is in seeing how systems may be part of, or may be consistent with, other systems. So, for example, views (something which shows data computed from a database and usually much less data or restricted data to ensure confidentiality etc) are a functor into the database specifications I showed in the lecture.

Q: what are some topics in CT that you <u>never used</u> in consulting?
e.g. did you use natural transformations anywhere?
Yes, natural transformations correspond to database updates (there are some details about keys that are needed but basically….). Naturality ensures that, for example, an insertion leaves the database unchanged apart from the insertion. Once one knows that correspondence they are also useful in further developing the theory, so they do have an important role.
So what haven't I used? Among things that are common category theory for applications, closedness (monoidal closed or cartesian closed) has not had a role to play (probably because our consultancies focus on data rather than computation).