Applied Compositional Thinking for Engineers (ACT4E)

Session 3 - Specialization

Questions & Answers

Class experience

Note: I just discovered that there are streams in Zulip that we are not subscribed to by default. I found the link to "Adjoint School" from Jonathan in the stream beyond-act4e.

AC: I see, we should be advertising the new channels. We will do after this lecture as there are more channels being created.

Q: Is it possible to have some exercises after each class to test our knowledge? (Not to be turned in or anything, maybe solutions would be posted later) (+3)

Q: To add on to this, I'm finding it a bit hard to keep track of the definitions and would also really like this.

AC: It's a good idea, but we are a bit short on resources. If somebody more advanced can suggest interesting exercises, we will put them in the notes.

Joan: I suggest you go to Spivak and Fong Seven Sketches in Compositionality, they are solved at the end of the book so you can self check. (JL: +1)

Do those exercises correspond to the syllabus in this class?

Joan: yes, very much, though they build up to more depth. But the basic ones do.

Thanks Joan. I suggest starting a channel on Zulip for this. Seems at least 1 other person would like this.

Adding on, could you post a list of which exercises could be solved after each class?

AC: Started channel #Exercises on Zulip.

Questions on lecture materials

Subcategories

Q: In the database example, the preimages are relations, right?

AC**:** Yes, in general they are a 1-to-many relation.

Q: What is the difference between 1 and 2?
Q: In the definition of subcategory, how is 2 independent of 1?

AC: 2 is subtly more specific. Probably 1 could be shortened to "all the objects of C are in D" to avoid confusion between 1 and 2.

Requestion: is it possible that in Set this subtlety is not there and 1b implies 2?

AC: final consensus is that indeed the 2 and 1 are partially redundant.

Q: Why does the definition of subcategory require that the compositions of morphisms in C be morphisms in D? If this were not the case, D would not be a category anyway!

AC: Yu could have *composition* defined differently in C and D, I think. But I don't have an example/counterexample. Please write down an example if you know it.

DS: Let C and D be given by

Ob C = Ob D={x,y,z}, Hom(x,y)={f}, Hom(y,z)={g}, and Hom(x,z)={h,i}. (and of course identities)

So note that every object of C is an object of D and every morphism of C is a morphism of D. But we can set it up so that their composites are different:

in C, we put f ; g = h

in D, we put f ; g = i.

Now of course that's crazy. And that's the point. Subcategories should not be so crazy.

Jonathan mentioned how you could have different group laws on the same set. This is much more reasonable!

JL: An additional remark, since we haven't yet introduced how to think of a group as a category. Here's the idea: if (G, o) is a group, we can view it as defining a category where there is just one homset, given by the set G, and just one object, which we call e.g. " * ". We take the group operation " o " as our category's composition operation. And this will define a category!

Q: All the examples of subcategories so far are somehow 'trivial' because they represent things that are a special case of a more general thing. Are there more interesting examples coming?

The category of Swiss mountains is interesting, eh? :) >_< :D

I guess there's no way out of this triviality. Being a "sub-" of something means to be somewhat different from the rest, i.e. somewhat special, i.e. the special case of something.

AC: that's right, subcategories always feel trivial. When we get to functors and embeddings of a category into another, we will have more fun examples.

Q: Is "the set that contains all sets" in the category Set?

AC: Depends on how you build mathematics. In the Zermelo-F. formalization the *collection* of all sets is not a set. Hence it is not an object in Set.

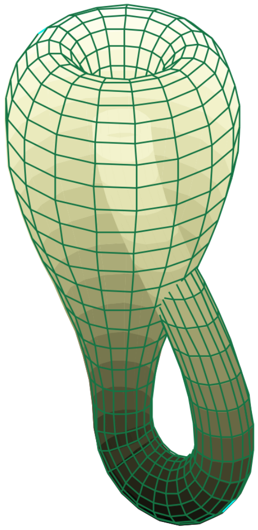
Q: Are the function morphisms a subset of the relation morphisms?

AC: Yes, it follows from the fact that Set is a subcategory of Rel.

Swiss mountains

Q: Silly question, but what is a manifold? Or smooth manifold? Or C^1-smooth manifold?

AC: good question. There is a long technical definition for "manifold" (*smooth* is assumed). It's basically what corresponds to the intuition of a smooth surface that you can travel on. (this surface could have n dimensions instead of 2). There are some fancy manifolds such as the Klein bottle:



Q: Is *p* a continuous variable? Or is L a finite set?

AC: Yes, p is continuous. Think of the actual Swiss mountain (but remove the trees because it's supposed to be a smooth manifold).

Q: wrt to the Concatenation: at point T1 we have y(1) and y(2) defined. Shouldn't be one excluded? For reference to this question see y(t) on page 12/17 of the slides

JL: The first path ends at the point the second path begins, so all is well I believe.

Q: In the example of the Berg category what would be the opposite category? Does it exist or does it have meaning to talk about the opposite category in that example?

JL: I don't think there is any concept of "inverse category"
Just to be specific. Do you mean opposite or inverse category?

I mean opposite category, sorry for the misunderstanding, I was not sure of the correct terminology.

Well then diving to the depths of the Mariana trench in the Pacific Ocean would be the opposite of climbing the mountains :) One of the main problems, the lack of oxygen is present in both categories..

However, my diving example is wrong, cause the objects must stay the same, it's just the morphisms which you need to reverse to get the opposite category.

AC: it's a mountain inhabited by Swiss people that can only travel backwards.

Counter-examples

Q: What would be an example of a non-category (that is close to a category) because an identity function does not exist?

GO: you can construct naively. Draw a graph representing a category but avoid to draw the identity arrow on one object. Rather artificial I agree but…

Q: So, more technically, what are examples of this [Semigroupoid - Wikipedia] ?

AC: Discrete-time linear systems in control theory. See previous Q&As from Session 2.

Other topics

Q: What are the ways category theory has been used in database programming languages (for example, is it possible to make some more expressive SQL?)

AC: so, relational databases are all about relations and they have very good mathematical semantics. There will be more in the book.

Regarding a categorical way to look at DBs, take a look at https://www.categoricaldata.net/

We probably will discuss this later after we introduce functors.

Q: In the category Rel it turns out, if I'm not wrong, that a morphism is also an object of the category. Do you have other examples of categories where that happens?
Right…;o)

The category of vector spaces and matrices maybe?

A matrix is not an object of Vect…

Matrices can be added, subtracted, there is a zero matrix. You can multiply with scalars.

I agree but one matrix alone is not a vector space and then not an object of Vect.

True! My mistake.

In any category where there is only one morphism between any pair of objects. Like a poset, or a lattice regarded as a category.

Maybe the category Hask, category of Haskell types. A function between types is also a type.

AC: I think a function *has* a type, but it *is not* a type. In that case, the homsets are types.
e.g. Take A, B types.
A function f: A -> B is a morphism between types A, B.
The hom-set Hom(A;B) is the type A -> B.
So the hom-set is an object of the category, but a morphism is not an object of the category.

Nice one. In Lisp a program is data.

AC: I think this works if you say: the objects of CLisp are S-expressions, the morphisms are also S-expressions. (this is different from Hask where the objects are types).

Q: A bit off-topic from Set and subcategories, but the database example leads me to ask: Is there a CT compatible or natural way of dealing with attributes or quantities of varying units?

AC: regarding varying units, the currency category **Curr** is such an example.
Typically you want to say that instead of R, you have many copies of R called R_g, R_m, R_K and so on (with all the basic units). Same thing as in **Curr**.