Applied Compositional Thinking for Engineers (ACT4E)

Session 1 - Transmutation

Questions & Answers

Meta-questions

Any updates from the doc from the last lecture? There were a lot of good ideas on both topics and organizing the Zulip?

AC: yes we updated it a bit, reorganized, then froze it:
https://applied-compositional-thinking.engineering/wp-content/uploads/2021/01/Session-0-Introduction-QA.pdf
AC: as for the implementation of the suggestions, we did some: calendar, lecture slides.
Working on the wiki.

Technical questions

Why is it wheels, then motor rather than motor, then wheels? Or are they the same?

AC: convention - will discuss at length later the choice of convention.

See also this video: https://vimeo.com/499223785

Q: How would brakes be modeled?

NM: I'm going to guess it would be: heat --brakes--> translational energy

Q. Why are the arrow directions reversed in car example vs currency exchange? In car arrow is from destination to source while in currency, it is from source currency to destination currency.

AC: the directions of the arrows don't need to be consistent with intuition about the semantics.

A: Arrows are just symbols representing relations. Any infix symbols can also be used to represent, such as @ $ % and of course reverse of arrow.

Note: It may be important to keep in mind that in mathematics functions don't "consume" anything. "Consume" in the sense that the input doesn't exist any more, after applying the function over the input.
Outside linear logic ;-)
yeah right :)

Q: Why does the direction of the arrow not matter? USD -> CHF != CHF -> USD and was shown with different options in the currency example.

AC: the direction matters once you choose the convention. But the convention does not matter. You could write everything else backward, nothing would change.

NM: As a bit of clarity while you can write everything backwards this doesn't mean that the transformation USD->CHF is the same as CHF->USD.

Q: What if a morphism corresponds to a function that is many to one? How can its reverse also be a morphism? Or are you saying that you can always model the morphisms in reverse, but the reverse morphisms may not represent simple inverses of the morphisms?

AC**:** In general if there is a morphism f: X -> Y, not necessarily there is a morphism Y -> X. Or if there is, it is not necessarily the "inverse" of the first morphism.

Q: So what does the slide mean when it says that the direction of the morphisms is arbitrary and you can always show the morphisms pointing the other way?

AC: if you have a diagram   X -- f → Y --- g → Z you can reverse all the arrows at the same time to get different arrows, say blue arrows: Z -- g → Y --- f → X
The information in the two diagrams are the same, just the convention changed.

Q: Just curious, where can I learn more about linear logic, for a beginner? (I know about “normal” logic, in the sense of one intro logic class) :)

A: Search for “Frank Pfenning” on YouTube.

A: You may want to start with programming languages(Rust, Haskell, Idris) that support linear types and learn from examples.

AC: Note: if you are not a programmer, linear types are not necessarily the best way to learn about linear logic.

Q: In the currency exchange, does the presence of a commission (>0) means that the arrows don’t have a strict inverse?

AC: well, there could be irrational currency exchangers that give you some bonus money.

Is there a tool in CT to map chains of routes (in the SF example) to different routes? For instance, if there is a shorter route between two places than route A then route B.

NM: I think this is related to adjoint functors, but you may need to define a different category

Q: in the definition of category, the concept “collection of objects” has always been a bit mysterious to me. Does it have a more fundamental definition? OK that it is not a naive set to avoid Russell’s paradox, but then what is it? It is still “bigger” than a set (e.g. the category of all small categories is fine)

(AC: personally I think in terms of "types" as in type theory. Obc is a type, Hom is a type constructor. But I guess people have different metaphors/ways to think about it)

VBM: there are a few ways of getting around this problem, for example to work with size restrictions on your categories (finite / small / locally small), or to work with universes (e.g. Grothendieck universes) to provide cardinality bounds on *sets* of morphisms and objects. It’s usually sufficient to take collection = sets :)
(Sets are also defined as “collections” in ZF, it’s the things you can do with them that matters - and so it is in category theory)

FJR: Engineers live in an intuitionist (maybe finitist/constructivist/ultrafinitist…) universe, so we can all just do sets :)

Maybe because the category's objects can be morphisms?

FJR: Doesn't that still leave us in a countable product of sets?

VBM: I may be misunderstanding the question, but what's important about a category is ultimately not what the nature of the objects are (or if they're elements of a set, or some more exotic collection), but how the morphisms relate them. For example, a finite category (finitely many objects, finitely many morphisms) can be cast as a finite set - but doing so forgets all about the distinction between objects and morphisms, and which morphisms map between which objects.

NM: One of the favourite things I've heard regarding category theory is that thinking about elements of an object is evil, It might help to think of a category of thermodynamics of gases if you aren't from software engineering since the objects are the different thermodynamic configurations (pressure, volume, number of particles, temperature) which have elements that are particular configurations of the atoms in these gases, and the morphisms are transformations you could do. However it is natural to only think about the thermodynamic quantities for these systems rather than the microscopic configuration.

Q: Wouldn't unitality follow directly from the definition of Identity morphism?

AC: you are probably thinking that "id" is already defined ,e.g. as identity of functions. The definition says that idx can be anything that satisfies the axiom.

Q: in the case of the electric car, do we think of the model as just a structural model (how the car is built) or we can make it into a dynamical model (how it works)?

AC: maybe a "functional model"? The function of the motor is to provide rotational motion to the axle. It requires electric power. (a function provided by a battery, etc.)

Q: So it would not be possible to express for example a law of conservation of energy (kinetik + battery)?

AC: yes but we need to put numbers on the quantities. So far this was a very cartoonish example.

Q: for me it is still not clear what the difference between a function and a morphism is? What is the main necessity to define a parallel system to the common functions and function compositions? With funny new symbols..

AC: In none of these examples the arrows were *functions*…

NM: there is a category of relations, where the objects are sets, but the morphisms tell you if they are related in some way (e.g. a relation could be if two triangles are similar/have the same angles).

Q: Followup: but they could have been expressed as functions? Currency exchange - linear transformations. What would be the main difference between morphism and a function? There is a good reason for almost the same notation and definition parts.

AC: (First, note that not all of these could be expressed as functions; we will have examples of much broader categories or more precise examples that clarify that there is no way express the morphisms as functions.)

There is a category of "Sets and Functions" which reproduces what you know about functions.

If you have a bunch of functions, do you gain something by recognizing the structure of a category?

Well, for example, a category is closed with respect to composition. For example, saying that "functions of the type x -> a x with a > 0" are a category (a subcategory of the category of sets and functions) also implies this additional information about being closed to composition.

Q: Follow-up: so then the theory comes from the meta functional level? From the place where relations are defined?

JL: Functions, relations, and many other things are all special cases of the concept of a morphism. Category theory is very abstract/general: the idea of a morphism is something which can have many different incarnations/realizations in various mathematical/applied contexts. We will continue to give many examples which should make this more clear with time.

Q: if the morphisms are not necessarily functions, how about (non-functional) relations?

JL: We will talk about this sort of thing in Session 2 !

Q: Are there collections of objects and morphisms that don't form a category? What is an example?

AC: They get called "morphisms" only if they are part of a category; I guess the question is "what are examples of points/arrows systems that don't form a category?"?

Yeah^, sorry I don't know the notation and terminology

NM: I think the answer here is that objects and morphisms don't form a category if the morphisms either: lack an identity for every object, don't include a morphism as a result of composing a (sensible) pair of morphisms, or morphism composition fails associativity.
Examples are:
- certain ways of defining magmas, quasigroups (single object, but morphisms are the elements of the magmas/quasigroups and composition is the binary operator)
- category where objects are memory allocation states and associated finite integer clock, and morphisms are state updates that never decrease the value of the clock. You can't compose

morphisms that combined would increase the clock by more than the total number of steps it could take and so don't always have composition. (Though my definition really requires a number of poor choices to be technically rigourous and can easily be fixed by just saying the clock overflows if it takes too long, but hopefully hints as to when/why things can break)

Q. Did you ever run into problems where associativity of composition (or corollaries based on the basic axioms) did not hold with the engineering side of things? How did you overcome these problems?

AC: Yes, for example, linear discrete proper dynamical systems characterized by three matrices A,B,C,
and x(t+1) = Ax(t) + Bu(t); y(t) = Cx(t) do not form a category because there are no identities.

NM: Here I think you need to include a zero time step which is your identity, by composition you'd have all positive integer time steps (assuming that your morphisms are updates of the variable pair {x(t), y(t)})

AC: you can fix it in many ways; however the point is that <u>inside the normal formalization</u>, in which composition is defined, you cannot find an identity.

NM: Is this the normal formalisation from engineering?

AC: yes, in control theory, this is the formalization for <u>linear</u> and <u>proper</u> discrete dynamical systems. If you take the more general class
x(t+1) = Ax(t) + Bu(t);
y(t) = Cx(t) + Du(t)
Then you do have an identity: A=B=C=0, D= 1.

However - you will have problems finding a way to "close the loop" (categorical "trace").

NM: what are your objects, it sounds like your morphisms are characterised by {A,B,C,D} but it sounds like u(t) is also part of the morphisms as well

AC: There are many ways to define the objects here. In the finite case, x,y,z have finite dimension.
So x : R^n, u : R^m, y: R ^ p.
The input to the system has dimension m and the output dimension p.
A mathematician would say then that the objects of this category are the natural numbers.
The system is a morphism in Hom_DDS(m;p).

If you care about more properties of dynamical systems, you might want to say that a linear system maps "linear subspaces of signals" to "linear subspaces of signals", and those would be the objects.

MC: Sorry, but how did you overcome the identity problem?

NM: If you are asking me it may be because I assumed the category has objects that are configurations of {x(t),y(t)} and morphisms are update steps. By composition you should have morphisms corresponding to multi-integer time steps, then the identity would be the zero time update
The category is defined by the following data:
- Objects are the set of all pairs of variables {x,y}
- Morphisms are all maps {x(t),y(t)} -> {x(t+1),y(t+1)} generated by inputs {A,B,C,u(t)}, and all maps that can be generated by concatenation of the prior maps (allowing for morphisms representing {x(t),y(t)} -> {x(t+n),y(t+n)} which may have different A,B,C at different time steps).
- The identity is the maps {x,y}->{x,y}
- Composition is as expected

Q: Why is it necessary to add a state for the definition of process? Is it possible to define them without a state space?

JL: The example we gave was one possible model/formalization of what one might call a “process”. There are certainly other ways to formalize this vague idea; though not all formalizations will necessarily match directly with concepts of category theory!

Q: Can the objects in a category also be morphisms? Or similarly, can a morphism produce another morphism?

JL: In general (a priori), there is no reason for this to be the case always. However, there are plenty of examples where one can view a given mathematical entity in different ways - in one way as an object of some category C and in another way as a morphism in some other category C’.

Q: How can we express associativity in engineering terms?

AC: "If you have a checklist to go through, the final outcome is independent of how you group the operations on the checklist as long as you maintain the order".

Q: Sometimes in the formulas there are arrows like ---> and sometimes like |---> with a perpendicular line at the end opposing the head of the arrow. Can you clarify?

AC: Yes. This is standard notation in math - no need to think about category theory for this one.

If I want to define the function that squares numbers, I could write

f : R → R+
    a |→ a^2

The first line is domain/codomain. "f" takes real numbers (R) to nonnegative real numbers (R+).
The second line is telling you the way this operation is done. "I take an a and map it to a^2".

Applied Category Theory vs engineering and other fields

Q: What is a real world engineering example of a category where Obj really expands to a collection instead of being a mere set?

NM: Is this question from a mathematician?

MD: No, I'm not a mathematician. SW engineer. It was said in the lecture that "it's good to go beyond a set and have a collection and, in fact in engineering, we often do it". I found this statement, especially the second part, interesting.

VBM: depends on what you mean by "collection". Usually this is taken as a vague stand-in for sets, when one wants to argue with elements (as with sets) but doesn't want to worry about foundations, or size issues. It can also be a vague stand-in for "sets with structure" or "types" in some type theory. Most of the time, it does no harm at all to just think of sets. In real world engineering examples the categories in question are most likely to have sets as their objects (rather than, say, proper classes) as these kind of foundational questions don't pop up "in the real world".
However, we can expect to see many examples where the collections of objects or, more importantly, the collection of morphisms between objects is not just a mere set, but a set with structure (e.g. an order relation, as for locally posetal categories).

RH: Correct me if I'm wrong but I believe that this situation will arise in homotopy type theory.

NM: I think this is the case (though not a mathematician), but one way to generalise away from sets is to consider higher categories, the simplest example is the category of categories where every object is a category (the morphisms are functors), this is a example of something called a 2-category. If I recall correctly a 1-category has objects and hom-sets are actually sets as we normally think about them.

VBM: It depends on how you approach homotopy types. The most fruitful way (IMO) is to model homotopy types as topological spaces modulo homotopy equivalence, in which case your hom collections are "equivalence classes of sets with additional structure".

Q: Perhaps off topic: I hear that CT can provide an alternative foundation of mathematics to set theory. Is there an analogous set of axioms like Zermelo-Fraenkel-Choice? Or what does this mean a bit more technically?

VBM: here's a link foundation of mathematics in nLab (ncatlab.org) to an article that lays this out well

Q: Rx{USD} is isomorphic to R, but we still need the x{USD} part to distinguish it from e.g. Rx{EUR}. Nice way of fixing units and distinguishing domains! Is this somehow standard? First time I see it!

NM: I feel like this is similar to how you have types in programming languages

AlexM: in programming we also say two functions (lambda x : x) and (lambda y : y) are the same even though written differently.

VBM: this question highlights a really important aspect of category theory: the difference between equality and isomorphism. For example, the sets A={a}, B={b}, and *={1} are not equal, but are certainly isomorphic (bijective). But note when we say “A and B are isomorphic sets” this is actually a statement of an equivalence of some objects in a certain category. Context is key!

AC: Not sure if the formalization is standard, but certainly there many isomorphic formalizations.

JL: An additional note - yes, exactly, we use the labels “USD”, etc., to distinguish different objects from one another. In this particular example, one can actually do away completely with the “R” part of the objects…. but this might seem confusing at first. In the lecture notes this is covered in a bit more detail.