

Re: Category Theory of Algorithms

Source: <http://coding.derkeiler.com/Archive/General/comp.theory/2007-04/msg00053.html>

- *From:* "the.theorist" <the.theorist@xxxxxxxxx>
 - *Date:* 11 Apr 2007 12:22:56 -0700
-

Everyone has given me some really great references, thank you all!

Jym wrote:

I guess one of the main difficulty is to get a precise definition of what is an algorithm, one on which everyone can agree.

Then, I'm not sure category theory will be the best suited tool to do the stuff...

Mitch wrote:

The closest I can imagine to what you are looking for is really in – programming language theory–, where you –do– compose separate pieces of (arbitrary) code with different operations, and there is a well studied (and continuing study) application of category theory to programming languages.

It's quite fascinating that the semantics of programming languages was touched on. One of the reasons that I wanted an algebra of algorithms is to study the relationship between the algorithm (which hasn't been formally defined) and its expression in a particular language. I thought that by attempting a formalization of the algorithm, as a mathematical object, this relationship could be made a little more clear. This is why I thought Category Theory might help, as a unifying lens.

(The Sapir–Whorf style relationship between what we think (algorithms) and what we say (programs) is what I'm aiming to go to grad school to study. Also how that relationship could help guide the creation of computer languages.)

Mitch wrote:

'Algorithms' (or anything with algorithm in it) usually refers to the study of design and analysis of specific problems: e.g. shortest paths, string matching, matrix multiplication, satisfiability of logical formulas. Take a particular problem, solve it.

Re: Category Theory of Algorithms

It really is unfortunate that I don't have any particular examples of what I mean by 'algebra of algorithms'.

It was the particularity of a solution to a problem that I wanted to get rid of. I was hoping that a study of algorithms could be made that would free each algorithm from its problem domain. Given that algorithms found in graph theory are useful for the practical world, it's human insight and experience that makes such a connection. I mean, there isn't any theory that would tell us where a particular algorithm might be applicable. My aim was to abstract the noting of algorithm so that it could be given clear, specific ties to all the rest of mathematics, connections that would be formally established, rather than ad hoc. I'd really find it nice if there was a system that would aid our insight into the connection between an algorithm and its various uses in the real world.

Jym wrote:

However, we're not currently really studying algorithms as mathematical objects in a "algebra of algorithms" sense. I guess such an algebra could be build with the right insight but theory of algorithms is quite inexistant (it is not even easy to agree on what is an algorithm, cf also works of Gurevich or Moskovakis) compared to theory of functions so it will probably take some times before such a "structured" achievement can be done.

....

I'm currently looking at a way to use a tensor algebra to study algorithm. It's only a wild idea currently even is some examples seem to work quite well...

Interesting that you mention Moskovakis, he was my set theory teacher, while I was an undergrad at UCLA.

I'd really like to see some of those examples, It might assist in clearing the fog.

Chris F Clark wrote:

Note that the "problems" aren't at that level either. People can write provably correct programs using simple algorithms and compositional methods today. However, most people don't do that.

Very good point. To get back to my originally stated intent: to find a means of moving us to a parallel paradigm.

I've got a half-baked thought that such a transition could be made if a language were to be invented where expressing one-self in a non-parallel fashion would be impossible, and the language would enforce this in its grammar. Such a language would prohibit operations that weren't idempotent, and would probably end up quite functional in nature. That way you don't have to work for parallelization, it would

Re: Category Theory of Algorithms

come naturally for that language.

Jamie Andrews wrote:

In order to create a calculus of algorithms, you need to have mathematical objects standing for algorithms. In order to do this, you have to formalize algorithms in some way. I can't see any way of formalizing them that would not end up as some kind of simple programming language. You would therefore end up in the well-studied area of category-theoretic foundations of programming languages.

At this point, I quite agree, It's not possible to create an algebra of algorithms, as I'd originally intended. Through this discussion (again, thank you all), it's become increasingly evident that algorithms themselves are simply too powerful to be reduced to some sort of algebra. In attempting to do so, you'd end up creating a mini-programming language, and wind up precisely where you started.

However, I haven't seen a formal proof that this is in fact the situation; such a proof would be a nice result to have. Perhaps an isomorphism can be made between algebras and some class in the complexity hierarchy, and then a demonstration that algorithms form a higher class in the hierarchy? or perhaps it's been proven already, and I'm simply unaware of it.

.