

# Re: Category Theory of Algorithms

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.theory/2007-04/msg00039.html>

---

- *From:* Chris F Clark <cfc@xxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* Tue, 10 Apr 2007 12:31:59 -0400
- 

the.theorist wrote:

...

See what I'm after is an algebra of algorithms. I'd like to know if it's possible to take some small set of simple algorithms, and by repeated application of some operators build up to more complex algorithms. An, Algebra or Calculus of Algorithms, if you will.

...

Patricia Shanahan <pats@xxxxxxx> wrote a good reply including:

....

If we could not build up more complex algorithms from simple ones, there would be little point in studying standard algorithms. Very few people are directly interested e.g. in finding the shortest path between two vertices in a graph. The algorithm for doing that is important because we can use it as a unit in programs that solve a real world problems.

So, the answer to your question is both "yes" and "no" (and the "no" is actually a "yes" also, as I'll get to).

For the first yes, we do have a "calculus" of combining algorithms (actually numerous ones). In one version, it's operations are sequencing, alteration, and looping. In another, we have recursion, composition, and alternation. Note that the adherents of functional programming (roughly) believe that one can live by the lambda calculus and algebras built on top of it. Gries in *\_The\_Science\_of\_Programming\_* did a pretty good exposition of a calculus and algebra for imperative programming based on the work of Dijkstra and Hoare.

And, that brings up the first "no" answer. The calculus isn't usually defined in terms of ands and ors. That being said, people have done algorithm calculi based on "and", "or", and composition. I recently read a article on building complex visitors, where they used those operators. However, "and" and "or" aren't always the best operators

## Re: Category Theory of Algorithms

for combining algorithms. So, you could probably build the calculus that you originally proposed. It just isn't likely to be revolutionary work, since we already have calculi at roughly that level.

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.