CHAPTER 1

Meta-meta and monads

Frequently there is no field of meta-meta-something; this is because meta-meta-something is a particular case of meta-something, so it is already included in your existing scope. This is a very strong kind of closure: you don't need layers upon layers ad infinitum: they all "collapse" to a single layer.

An abstraction of this structure in math is called a *monad*; I illustrate and define.

1. Examples

As an example (this was my motivating example), a common sophisticated technique in computer programming is to write a program that writes a program, and there are programs that can be written by a computer that no human could write, simply because they are so long: you couldn't write them if you typed your whole life. However, there is no theoretical reason to write a program that writes a program that writes a program: there is no class of computer-computer-writable programs. Why? Because given a computer-computer-writable program (program A writes program B that writes program C), have program A' write program B *and* an interpreter for the underlying language, then execute program B, thus outputing program C itself. (Note that I'm using Turing completeness.)

For practical reasons you might write a program to write a program to write a program (and hopefully avoid getting confused in the process), but this doesn't yield anything essentially new.

This meta-programming is often for flexibility; another common example is the programs `lex` and `yacc`, which, given a grammar, output C code that parses code written in that grammar: this allows writing in a simple domain-specific language.

More generally, meta-mathematics is itself mathematics, and meta-philosophy is itself philosophy. Further, meta-science is mathematics (the structure of science is math), so meta-meta-science is meta-math, which is again math.

As a meta-Hegelian example, philosophy of history isn't history (it's philosophy), but philosophy of philosophy of history is philosophy (note that it's not philosophy of history though: you have to choose a broad enough notion in order to be "closed", here "philosophy").

A frequent source of monads is adjunctions: A of B of A is a kind of A, so (A of B) of (A of B) is an (A of B). As a personal example, my "structure and eschatology of math" is math of philosophy of math: applying digraphs and AIT to the philosophy of math.

Similarly (and amusingly), philosophy of history of philosophy of history is (a kind of) philosophy of history. Hegel would be delighted. Note that I'm using right-associativity: I mean philosophy of (history of (philosophy of (history))).

The deeper point here is that once you abstract a level, in studying this abstract level (doing meta-meta-something), you are insulated from the underlying: philosophy of philosophy of X is philosophy, whatever X may be.

2. What is a monad?

A simple example of a monad is "lists": given a set (an alphabet), you can consider lists (say of finite length) with terms from that set. You can also consider lists of lists, and lists of lists of lists, etc. Now a list of lists yields a list (of the basic elements), by concatenating the entries; this is the key operation. For instance the list of lists $((a,b),(c,d))$ yields the list (a,b,c,d); syntactically, by erasing the internal parentheses.

A key property of this process is that it is "associative": given a list of lists of lists, you can make this into a list in two ways: by thinking of it as a (list of lists) of lists or as a list of (lists of lists): concatenating at the outside or inside level first, then at the remaining level. These yield the same result: you're just erasing the parentheses in a different order. Starting with $(((a,b),(c,d)),((e,f),(g,h)))$, the first yields $((a,b),(c,d),(e,f),(g,h))$ while the second yields $((a,b,c,d),(e,f,g,h))$, and they both then yield (a,b,c,d,e,f,g,h).

The other property is that there is an empty list; this is technically convenient. A monad in category theory abstracts this via functors and natural transforms.

It's not a coincidence that this sort of structure should occur in category theory, as a key interest in category theory is exactly this sort of closure.

3. Other topologies

This "lack of meta-meta" is common but not universal: the theory of a field may be a different field, then the theory of that field may be another field again – or you may need to progress in several stages, as in boot-strapping: to write a C compiler from scratch, you'd generally write an assembler in machine code, write a very simple C compiler in assembler, write a more complex C compiler in simple C, and write a full C compiler in this more complex C subset – even though you could theoretically have written the full C compiler in machine code, you couldn't practically.

On the other hand, even nicer is when meta-X is itself X (as in math and philosophy; I call these "meta-closed fields"); in monadicity we're instead discussing meta-meta-X to meta-X, which is weaker.

There are some problems with such meta-closed fields: they are almost inevitably complicated, as they must contain all their own structure. Thus there can be pragmatic reasons for preferring a field where X, meta-X, meta-meta-X, and so forth are all distinct. See more in "Constrained systems are more structured".