



The First Law of Complexodynamics

A few weeks ago, I had the pleasure of attending FQXi's [Setting Time Aright](#) conference, part of which took place on a cruise from Bergen, Norway to Copenhagen, Denmark. (Why aren't theoretical computer science conferences ever held on cruises? If nothing else, it certainly cuts down on attendees sneaking away from the conference venue.) This conference brought together physicists, cosmologists, philosophers, biologists, psychologists, and (for some strange reason) one quantum complexity blogger to pontificate about the existence, directionality, and nature of time. If you want to know more about the conference, check out Sean Carroll's *Cosmic Variance* posts [here](#) and [here](#).

Sean also delivered the [opening talk](#) of the conference, during which (among other things) he asked a beautiful question: *why does "complexity" or "interestingness" of physical systems seem to increase with time and then hit a maximum and decrease, in contrast to the entropy, which of course increases monotonically?*

My purpose, in this post, is to sketch a possible answer to Sean's question, drawing on concepts from Kolmogorov complexity. If this answer has been suggested before, I'm sure someone will let me know in the comments section.

First, some background: we all know the [Second Law](#), which says that the *entropy* of any closed system tends to increase with time until it reaches a maximum value. Here "entropy" is slippery to define—we'll come back to that later—but somehow measures how "random" or "generic" or "disordered" a system is. As Sean points out in his wonderful book [From Eternity to Here](#), the Second Law is *almost* a tautology: how could a system *not* tend to evolve to more "generic" configurations? if it didn't, those configurations wouldn't *be* generic! So the real question is not why the entropy is increasing, but why it was ever low to begin with. In other words, why did the universe's initial state at the big bang contain so much order for the universe's subsequent evolution to destroy? I won't address that celebrated mystery in this post, but will simply take the low entropy of the initial state as given.

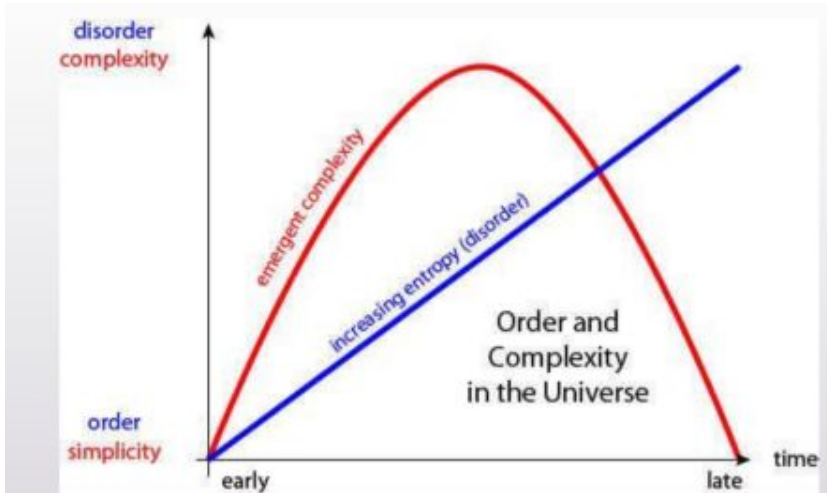
The point that interests us is this: even though isolated physical systems get monotonically more entropic, they *don't* get monotonically more "complicated" or "interesting." Sean didn't define what he meant by "complicated" or "interesting" here—indeed, defining those concepts was part of his challenge—but he illustrated what he had in mind with the example of a coffee cup. Shamelessly ripping off his slides:

Entropy increases.
Complexity first increases, then decreases.



low entropy medium entropy high entropy
low complexity high complexity low complexity

Entropy increases monotonically from left to right, but intuitively, the “complexity” seems highest in the *middle* picture: the one with all the tendrils of milk. And same is true for the whole universe: shortly after the big bang, the universe was basically just a low-entropy soup of high-energy particles. A googol years from now, after the last black holes have sputtered away in bursts of Hawking radiation, the universe will basically be just a *high*-entropy soup of *low*-energy particles. But today, in between, the universe contains interesting structures such as galaxies and brains and hot-dog-shaped novelty vehicles. We see the pattern:



Is there a law of thermodynamics governing the evolution of complexity?

In answering Sean’s provocative question (whether there’s some “law of complexodynamics” that would explain his graph), it seems to me that the challenge is twofold:

1. Come up with a plausible formal definition of “complexity.”
2. Prove that the “complexity,” so defined, is large at intermediate times in natural model systems, decreasing close to zero at the initial time and close to zero at late times.

Loading [MathJax]/extensions/MathMenu.js

To clarify: it's not hard to explain, at least at a handwaving level, why the complexity should be close to zero at the initial time. It's because we assumed the *entropy* is close to zero, and entropy plausibly gives an upper bound on complexity. Nor is it hard to explain why the complexity should be close to zero at late times: it's because the system reaches equilibrium (i.e., something resembling the uniform distribution over all possible states), which we're essentially *defining* to be simple. At intermediate times, neither of those constraints is operative, and therefore the complexity *could* become large. But *does* it become large? How large? How could we predict? And what kind of "complexity" are we talking about, anyway?

After thinking on and off about these questions, I now conjecture that they can be answered using a notion called [sophistication](#) from the theory of [Kolmogorov complexity](#). Recall that the *Kolmogorov complexity* of a string x is the length of the shortest computer program that outputs x (in some Turing-universal programming language—the exact choice can be shown not to matter much). Sophistication is a more ... well, sophisticated concept, but we'll get to that later.

As a first step, let's use Kolmogorov complexity to define *entropy*. Already it's not quite obvious how to do that. If you start, say, a cellular automaton, or a system of billiard balls, in some simple initial configuration, and then let it evolve for a while according to dynamical laws, visually it will look like the entropy is going up. But if the system happens to be *deterministic*, then mathematically, its state can always be specified by giving (1) the initial state, and (2) the number of steps t it's been run for. The former takes a constant number of bits to specify (independent of t), while the latter takes $\log(t)$ bits. It follows that, if we use Kolmogorov complexity as our stand-in for entropy, then the entropy can increase at most *logarithmically* with t —much slower than the linear or polynomial increase that we'd intuitively expect.

There are at least two ways to solve this problem. The first is to consider probabilistic systems, rather than deterministic ones. In the probabilistic case, the Kolmogorov complexity really does increase at a polynomial rate, as you'd expect. The second solution is to replace the Kolmogorov complexity by the *resource-bounded Kolmogorov complexity*: the length of the shortest computer program that outputs the state *in a short amount of time* (or the size of the smallest, say, depth-3 circuit that outputs the state—for present purposes, it doesn't even matter much what kind of resource bound we impose, as long as the bound is severe enough). Even though there's a computer program only $\log(t)$ bits long to compute the state of the system after t time steps, that program will typically use an amount of *time* that grows with t (or even faster), so if we rule out sufficiently complex programs, we can again get our program size to increase with t at a polynomial rate.

OK, that was entropy. What about the thing Sean was calling "complexity"—which, to avoid confusion with other kinds of complexity, from now on I'm going to call "complexropy"?

For this, we're going to need a cluster of related ideas that go under names like sophistication, Kolmogorov structure functions, and algorithmic statistics. The backstory is that, in the 1970s (*after* introducing Kolmogorov complexity), Kolmogorov made an observation that was closely related to Sean's observation above. A uniformly random string, he said, has close-to-maximal Kolmogorov complexity, but it's also one of the *least* "complex" or "interesting" strings imaginable. After all, we can describe essentially everything you'd ever want to know about the string by saying "it's random"! But is there a way to formalize that intuition? Indeed there is.

First, given a set S of n -bit strings, let $K(S)$ be the number of bits in the shortest computer program that outputs the elements of S and then halts. Also, given such a set S and an element x of S , let $K(x|S)$ be the length of the shortest program that outputs x , given an oracle for testing membership in S . Then we can let the *sophistication* of x , or $\text{Soph}(x)$, be the smallest possible value of $K(S)$, over all sets S such that

1. $x \in S$ and
2. $K(x|S) \geq \log_2(|S|) - c$, for some constant c . (In other words, one can distill all the “nonrandom” information in x just by saying that x belongs to S .)

Intuitively, $\text{Soph}(x)$ is the length of the shortest computer program that describes, not necessarily x itself, but a set S of which x is a “random” or “generic” member. To illustrate, any string x with small Kolmogorov complexity has small sophistication, since we can let S be the singleton set $\{x\}$. However, a uniformly-random string *also* has small sophistication, since we can let S be the set $\{0,1\}^n$ of all n -bit strings. In fact, the question arises of whether there are *any* sophisticated strings! Apparently, after Kolmogorov raised this question in the early 1980s, it was answered in the affirmative by Alexander Shen (for more, see [this paper](#) by Gács, Tromp, and Vitányi). The construction is via a diagonalization argument that’s a bit too complicated to fit in this blog post.

But what does any of this have to do with coffee cups? Well, at first glance, sophistication seems to have exactly the properties that we were looking for in a “complexity” measure: it’s small for both simple strings *and* uniformly random strings, but large for strings in a weird third category of “neither simple nor random.” Unfortunately, as we defined it above, sophistication still doesn’t do the job. For deterministic systems, the problem is the same as the one pointed out earlier for Kolmogorov complexity: we can always describe the system’s state after t time steps by specifying the initial state, the transition rule, and t . Therefore the sophistication can never exceed $\log(t)+c$. Even for probabilistic systems, though, we can specify *the set $S(t)$ of all possible states* after t time steps by specifying the initial state, the probabilistic transition rule, and t . And, at least assuming that the probability distribution over $S(t)$ is uniform, by a simple counting argument the state after t steps will almost always be a “generic” element of $S(t)$. So again, the sophistication will almost never exceed $\log(t)+c$. (If the distribution over $S(t)$ is nonuniform, then some technical further arguments are needed, which I omit.)

How can we fix this problem? I think the key is to bring computational resource bounds into the picture. (We already saw a hint of this in the discussion of entropy.) In particular, suppose we define the complexity of an n -bit string x to be something like the following:

the number of bits in the shortest computer program that runs in $n \log(n)$ time, and that outputs a nearly-uniform sample from a set S such that (i) $x \in S$, and (ii) any computer program that outputs x in $n \log(n)$ time, given an oracle that provides independent, uniform samples from S , has at least $\log_2(|S|)-c$ bits, for some constant c .

Here $n \log(n)$ is just intended as a concrete example of a complexity bound: one could replace it with some other time bound, or a restriction to (say) constant-depth circuits or some other weak model of computation. The motivation for the definition is that we want *some* “complexity” measure that will assign a value close to 0 to the first and third coffee cups in the picture, but a large value to the second coffee cup. And thus we consider the length of the shortest efficient computer program that outputs, not necessarily the target

Loading [MathJax]/extensions/MathMenu.js

string x itself, but a sample from a probability distribution D such that x is not efficiently compressible with respect to D . (In other words, x looks to any efficient algorithm like a “random” or “generic” sample from D .)

Note that it’s essential for this definition that we imposed a computational efficiency requirement in *two* places: on the sampling algorithm, and *also* on the algorithm that reconstructs x given the sampling oracle. Without the first efficiency constraint, the complexentropy could never exceed $\log(t)+c$ by the previous argument. Meanwhile, without the second efficiency constraint, the complexentropy *would* increase, but then it would probably keep right on increasing, for the following reason: a time-bounded sampling algorithm wouldn’t be able to sample from *exactly* the right set S , only a reasonable facsimile thereof, and a reconstruction algorithm with *unlimited* time could probably then use special properties of the target string x to reconstruct x with fewer than $\log_2(|S|)-c$ bits.

But as long as we remember to put computational efficiency requirements on *both* algorithms, I *conjecture* that the complexentropy will satisfy the “First Law of Complexodynamics,” exhibiting exactly the behavior that Sean wants: small for the initial state, large for intermediate states, then small again once the mixing has finished. I don’t yet know how to prove this conjecture. But crucially, it’s *not* a hopelessly open-ended question that one tosses out just to show how wide-ranging one’s thoughts are, but a relatively-bounded question about which actual theorems could be proved and actual papers published.

If you want to do so, the first step will be to “instantiate” everything I said above with a particular model system and particular resource constraints. One good choice could be a discretized “coffee cup,” consisting of a 2D array of black and white pixels (the “coffee” and “milk”), which are initially in separated components and then subject to random nearest-neighbor mixing dynamics. (E.g., at each time step, we pick an adjacent coffee pixel and milk pixel uniformly at random, and swap the two.) Can we show that for such a system, the complexentropy becomes large at intermediate times (intuitively, because of the need to specify the irregular *boundaries* between the regions of all-black pixels, all-white pixels, and mixed black-and-white pixels)?

One could try to show such a statement either theoretically or empirically. Theoretically, I have no idea where to begin in proving it, despite a clear intuition that such a statement should hold: let me toss it out as a wonderful (I think) open problem! At an empirical level, one could simply try to *plot* the complexentropy in some simulated system, like the discrete coffee cup, and show that it has the predicted small-large-small behavior. One obvious difficulty here is that the complexentropy, under any definition like the one I gave, is almost certainly going to be intractable to compute or even approximate. However, one could try to get around that problem the same way many others have, in empirical research inspired by Kolmogorov complexity: namely, by using something you *can* compute (e.g., the size of a gzip compressed file) as a rough-and-ready substitute for something you *can’t* compute (e.g., the Kolmogorov complexity $K(x)$). In the interest of a full disclosure, a wonderful MIT undergrad, Lauren Oullette, recently started a research project with me where she’s trying to do exactly that. So hopefully, by the end of the semester, we’ll be able to answer Sean’s question at least at a physics level of rigor! Answering the question at a math/CS level of rigor could take a while longer.