Hunter Adams

Are black holes evidence of a simulation? Is sleeping?

Protagonist: I have a problem with my computer program.

Friend: Go on.

Protagonist: I'm working on a project that requires I store arbitrarily large amounts of complexity in a computer with rather small amount of energy.

Friend: You want to store infinite amounts of data in a computer? That's impossible.

Protagonist: Not exactly. I want to store infinite amounts of complexity, but using as little data as possible. To give you a simple example of what I'm talking about, suppose that I wanted to store the number 1267650600228229401496703205376. Instead of storing that number, I could instead store the *equation* 2^{100}. The equation generates the much longer number, but can be stored in the computer using fewer bits (depending how I encode it).

Friend: Ah, that's much more tractable. I can think of a few examples of mathematical structures that are *infinitely* complex, and that have generating functions which are extremely compact. The Mandelbrot Set comes to mind. If you're simply trying to store infinite complexity compactly, solutions exist. Are you trying to store information which generates complexity of some specific and different variety?

Protagonist: Yes. I want for the emergent complex system to have the property that, as the system develops, its complexity consolidates to localized places within that system. This is in contrast to the other infinitely complex mathematical structures that you described, like the Mandelbrot Set. The Mandelbrot Set is indeed infinitely complex, but that complexity is distributed around its edge. As you run more and more iterations of the function which generates the Mandelbrot Set, you generate more and more complexity. But that complexity never *moves*, it only describes finer and finer twists and turns of the stationary edges of the set.

Protagonist (continued): Imagine if, as you ran more and more iterations of the Mandelbrot Set, the complexity began to gather in specific places. Some regions of the Set would become less complex, while others would consolidate increasing amounts of complexity. That's the kind of system that I'm after. One which generates infinite complexity, but which also continues to consolidate that complexity into smaller and smaller spaces, and that does so in interesting ways.

Friend: And you wrote a program to try to find such a generating function?

Protagonist: Yes, nothing very complicated. The program instantiates a bunch of simulations that contain fundamental objects with a randomized set of properties, and each simulation has a randomized set of rules for how those objects can combine and interact.

Friend: So the actual data that you're storing is just the properties of these fundamental objects, and a description of the rules which describe how those objects interact?

Protagonist: Yes, and then I'm allowing each system to evolve in the hopes of finding a particular set of properties/rules that generate infinite complexity, and which consolidate that complexity over time. Each simulation is run for a while, then evaluated based on how well it has generated and consolidated complexity. Those that do a good job are combined to try to create even better simulations, and those that do poorly are discarded.

Friend: Have any of the simulations generated anything interesting?

Protagonist: Yes, but the problem is evaluating the simulations. In order to see how well they generate and consolidate complexity, I need to run all the rules on all the objects in each simulation. As the simulations consolidate complexity, there are particular regions of the generated system that take an impractical amount of time to compute. It's promising that the simulations are generating and consolidating complexity, but my computer just isn't fast enough to evolve it to the next generation. So, I'm a bit stuck.

Friend: Do these regions ever spawn new regions of complexity? Or do they only suck in complexity that never escapes?

Protagonist: They only suck in complexity that never escapes.

Friend: Well, just as an experiment, perhaps you could skirt the problem. Draw a boundary around the region. If anything enters, just delete it. That way you can continue to evolve your simulations without spending the computation in those regions that slow the system down.

Protagonist: That's a good idea. I'll put a black dot over those regions. Stuff can enter the black dot region, but nothing may ever escape. Once anything enters that region, I'll stop running any computation on it. That will allow for me to see if complexity is consolidated in any other interesting ways in later generations, without slowing the system down by continuing to run computation in the black dot regions.

The Protagonist updates the code. As he does so, his Friend makes idle conversation.

Friend: Did you see that recent image of the black hole that scientists gathered?

Protagonist: Yeah, very cool.

Protagonist finishes his updates, and reruns the simulation.

Friend: Is it behaving any differently now?

Protagonist: Yes, it's no longer getting stuck running computation in the black dot regions. Complexity is being consolidated in different ways now. I'm now seeing systems that not only suck complexity into smaller and smaller spaces, but that also spawn additional complexity in the regions around them.

Friend: That sounds interesting.

Protagonist: Yes, but now I'm already running into the same issue as before. The areas where complexity has been consolidated now take an impractical amount of time to compute. And I can't use the same strategy as before. Because these particular systems seem to add complexity to their environments, it's impossible to draw a clean boundary around them.

Friend: Perhaps you could schedule computation for those systems? Briefly pause updating some of those systems so that you can update the others, then switch which ones are being updated?

Protagonist: Ah, that's a good idea. I can add a simple scheduler that pauses some systems while running others, and that then switches which are being paused and which are being run. That way I can keep the simulation running for longer without it taking an impractical amount of time to continue to update it.

Friend: Let me know how that goes. I'm going to sleep.

Protagonist: Me too. My family overseas is just waking up. I'm going to chat with them and then go to bed.