Quantifying the Rise and Fall of Complexity in Closed Systems: The Coffee Automaton

Scott Aaronson[*] Sean Carroll[†] Lauren Ouellette[‡]

Abstract

In contrast to *entropy*, which increases monotonically, the "complexity" or "interestingness" of closed systems seems intuitively to increase and then decrease: for example, our universe lacked complex structures at the Big Bang and will also lack them after it reaches thermal equilibrium. This paper makes an initial attempt to quantify this pattern. As a model system, we use a simple, two-dimensional cellular automaton that simulates the mixing of two liquids ("coffee" and "cream"). A plausible complexity measure is then the Kolmogorov complexity of a coarse-grained approximation of the automaton's state. We study this complexity measure, and show analytically that it never becomes large when the liquid particles are non-interacting. By contrast, when the particles *do* interact, we give numerical evidence that the complexity reaches a maximum comparable to the "coffee cup's" horizontal dimension. We raise the problem of proving this behavior analytically.

1 Introduction

Imagine a cup of coffee into which cream has just been poured. At first, the coffee and cream are separated. Over time, the two liquids diffuse until they are completely mixed. If we consider the coffee cup a closed system, we can say that its *entropy* is increasing over time, in accordance with the second law of thermodynamics. At the beginning, when the liquids are completely separated, the system is in a highly ordered, low-entropy state. After time has passed and the liquids have completely mixed, all of the initial structure is lost; the system has high entropy.

Just as we can reason about the disorder of the coffee cup system, we can also consider its "complexity." Informally, by complexity we mean the amount of information needed to describe everything "interesting" about the system. At first, when the cream has just been poured into the coffee, it is easy to describe the state of the cup: it contains a layer of cream on top of a layer of coffee. Similarly, it is easy to describe the state of the cup after the liquids have mixed: it contains a uniform mixture of cream and coffee. However, when the cup is in an intermediate state—where the liquids are mixed in some areas but not in others—it seems more difficult to describe what the contents of the cup look like.

Thus, it appears that the coffee cup system starts out at a state of low complexity, and that the complexity first increases and then decreases over time. In fact, this rising-falling pattern of complexity seems to hold true for many closed systems. One example is the universe itself.

[*] MIT. Email: aaronson@csail.mit.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0844626. Also supported by a DARPA YFA grant, an NSF STC grant, a TIBCO Chair, and a Sloan Fellowship.

[†] Caltech. Email: seancarroll@gmail.com. INSERT FUNDING INFO

[‡] This work was done while the author was a student at MIT. Email: louelle@alum.mit.edu.

The universe began in a low-entropy, low-complexity state with the Big Bang, and is predicted to end in the high-entropy, low-complexity state of heat death; but in between, complex structures such as planets, stars, and galaxies have developed. There is no general principle that quantifies and explains the existence of high-complexity states at intermediate times in closed systems. It is the aim of this work to explore such a principle, both by developing a more formal definition of "complexity," and by running numerical experiments to measure the complexity of a simulated coffee cup system.

2 Background

Before discussing how to define "complexity," let's start with the simpler question of how to define entropy in a discrete dynamical system. There are various definitions of entropy that are useful in different contexts. Physicists distinguish between the Boltzmann and Gibbs entropies of physical systems. (There is also the phenomenological thermodynamic entropy and the quantum-mechanical von Neumann entropy, neither of which are relevant here.) The Boltzmann entropy is an objective feature of a microstate. We imagine coarse-graining the space of microstates into equivalence classes, so that each microstate x_a is an element of a unique macrostate X_A. The volume W_A of the macrostate is just the number of associated microstates $x_a \in X_A$. Then the Boltzmann entropy of a microstate x_a is the normalized logarithm of the volume of the associated macrostate:

$$S_{\text{Boltzmann}}(x_a) := k_B \log W_A,$$

where k_B is Boltzmann's constant (which we can set equal to 1). The Boltzmann entropy depends on a choice of coarse-graining, but is independent of our knowledge of the system; in particular, it can be nonzero even when we know the exact microstate. The Gibbs entropy (which was also studied by Boltzmann), in contrast, refers to a distribution function $\rho(x)$ over the space of microstates, which can be thought of as characterizing our ignorance of the exact state of the system. It is given by

$$S_{\text{Gibbs}}[\rho] := -\sum_x \rho(x) \log \rho(x).$$

In probability theory, communications, information theory, and other areas, the Shannon entropy of a probability distribution $D = (p_x)_x$ is the expected number of random bits needed to output a sample from the distribution:

$$H(D) := -\sum_x p_{\log} p_x.$$

We see that this is essentially equivalent to the Gibbs entropy, with a slight change of notation and vocabulary.

Finally, in computability theory, the entropy of an n-bit string x is often identified with its *Kolmogorov complexity* $K(x)$: the length of the shortest computer program that outputs x.[1] Strings that are highly patterned—meaning low in disorder—can be described by a short program that takes advantage of those patterns. For example, a string consisting of n ones could be output by a short program which simply loops n times, printing '1' each time. Conversely, strings which

[1] A crucial fact justifying this definition is that switching from one (Turing-universal) programming language to another changes $K(x)$ by at most an additive constant, independent of x. The reason is that in one language, we can always just write a compiler or interpreter for another language, then specify x using the second language. Also, throughout this paper, we will assume for convenience that the program receives x's length n as input. This assumption can change $K(x)$ by at most an additive $O(\log n)$ term.

have little regularity cannot be compressed in this way. For such strings, the shortest program to output them might simply be one that hard-codes the entire string.

Fortunately, these notions of entropy are closely related to each other, so that one can often switch between them depending on convenience. The Gibbs and Shannon entropies are clearly equivalent. The Boltzmann entropy is equivalent to the Gibbs entropy under the assumption that the distribution function is flat over microstates within the given macrostate, and zero elsewhere–*i.e.*, given the knowledge of the system we would actually obtain via macroscopic observation. For a computable distribution D over n-bit strings, the Kolmogorov complexity of a string sampled from D tends to the entropy of D [6].

Despite these formal connections, the three kinds of entropy are calculated in very different ways. The Boltzmann entropy is well-defined once a specific coarse-graining is chosen. To estimate the Shannon entropy $H(D)$ of a distribution D (which we will henceforth treat as identical to the corresponding Gibbs entropy), one in general requires knowledge of the entire distribution D, which could potentially require exponentially many samples from D. At first glance, the Kolmogorov complexity $K(x)$ seems even worse: it is well-known to be *uncomputable* (in fact, computing $K(x)$ is equivalent to solving the halting problem). On the other hand, in practice one can often estimate $K(x)$ reasonably well by the compressed file size, when x is fed to a standard compression program such as `gzip`. And crucially, unlike Shannon entropy, Kolmogorov complexity is well-defined even for an individual string x. For these reasons, we chose to use $K(x)$ (or rather, a computable approximation to it) as our estimate of entropy.

Of course, *none* of the three measures of entropy capture "complexity," in the sense discussed in Section 1. Boltzmann entropy, Shannon entropy, and Kolmogorov complexity are all maximized by "random" or "generic" objects and distributions, whereas a complexity measure should be low both for "simple" objects *and* for "random" objects, and large only for "interesting" objects that are neither simple nor random. This issue has been extensively discussed in the complex systems and algorithmic information theory communities since the 1980s. We are aware of four quantitative ideas for how to define "complexity" or "interestingness" as distinct from entropy. While the definitions look extremely different, it will turn out happily that they are all related to one another, much like with the different definitions of entropy.

2.1 Coarse Entropy

The first notion, and arguably the most "obvious," is *coarse entropy*. By the coarse entropy of an object x, we mean $H(f(x))$, where H is any of the entropy measures discussed previously, and f is some "denoising" or "smearing-out" function—that is, a function that attempts to remove the "incidental" or "random" information in x, leaving only the "interesting, non-random" information. For example, if x is a bitmap image, then $f(x)$ might simply be a blurred version of x.[2]

Coarse entropy has two immense advantages. First, it is simple: it directly captures the intuition that we want something *like* entropy, but that leaves out "incidental" information. Second, we can plausibly hope to *compute* (or rather, approximate) coarse entropy: we need "merely" solve the problems of computing H and f. It's because of these advantages that the complexity measure we ultimately adopt for our experiments will be an approximate variant of coarse entropy.

On the other hand, coarse entropy also has a large disadvantage: namely, the apparent arbitrariness in the choice of the denoising function f. Who decides which information about x is

[2] Note that coarse entropy is *different* from "coarse-grained entropy"; the latter term is sometimes used in physics as a synonym for Boltzmann entropy. By contrast, coarse entropy is an attempt to quantify *complexity*, and need *not* monotonically increase in closed systems.

"interesting," and which is "incidental"? Won't f depend, not only on the type of object under study (bitmap images, audio recordings, etc.), but even more worryingly, on the prejudices of the investigator? For example, suppose we choose f to blur out details of an image that are barely noticeable to the human eye. Then will studying the time-evolution of $H(f(x))$ tell us anything about x itself, or only about various quirks of the human visual system? Ideally, we'd like our complexity measure to *tell us* what the distinction between "random" and "non-random" information consists of, rather than having to decide ourselves on a case-by-case basis. This motivates an examination of some alternative complexity measures.

2.2 Sophistication

The second notion—one that originates in work of Kolmogorov himself—is *sophistication*. Roughly speaking, sophistication seeks to generalize Kolmogorov complexity to capture only the non-random information in a string—while using Kolmogorov complexity itself to define what is meant by "non-random." Given an n-bit string x, let a *model* for x be a set $S \subseteq \{0,1\}^n$ such that $x \in S$. Let $K(S)$ be the length of the shortest program that enumerates the elements of S, in any order (crucially, the program must halt when it is done enumerating the elements). Also, let $K(x|S)$ be the length of the shortest program that outputs x given as input a description of S. Then we can consider x to be a "generic" element of S if $K(x|S) \geq \log_2 |S| - c$ for some small constant c. This means intuitively that S is a "maximal" model for x: one can summarize all the interesting, non-random properties of x by simply saying that $x \in S$.

Now the *c-sophistication* of x or $\operatorname{soph}_c(x)$, defined by Koppel [5], is the minimum of $K(S)$ over all models S for x such that $K(S) + \log_2 |S| \leq K(x) + c$. (The optimal such S is said to "witness" $\operatorname{soph}_c(x)$.) In words, $\operatorname{soph}_c(x)$ is the smallest possible amount of "non-random" information in a program for x that consists of two parts—a "non-random" part (specifying S) and a "random" part (specifying x within S)—assuming the program is also near-minimal. We observe the following:

(i) $\operatorname{soph}_c(x) \leq K(x) + O(1)$, since we can always just take $S = \{x\}$ as our model for x.

(ii) *Most* strings x satisfy $\operatorname{soph}_c(x) = O(1)$, since we can take $S = \{0,1\}^n$ as our model for x.

(iii) If S witnesses $\operatorname{soph}_c(x)$, then $\log_2 |S| \leq K(x) - K(S) + c \leq K(x|S) + c$, meaning that x must be a "generic" element of S.

It can be shown (see Gács, Tromp, and Vitányi [4] or Antunes and Fortnow [1]) that there do exist highly "sophisticated" strings x, which satisfy $\operatorname{soph}_c(x) \geq n - c - O(\log n)$. Interestingly, the proof of that result makes essential use of the assumption that the program for S halts, after it has finished listing S's elements. If we dropped that assumption, then we could always achieve $K(S) = O(\log n)$, by simply taking S to be the set of all $y \in \{0,1\}^n$ such that $K(y) \leq K(x)$, and enumerating those y's in a dovetailing fashion.

Recently, Mota et al. [7] studied a natural variant of sophistication, in which one only demands that S be a maximal model for x (i.e., that $K(x|S) \geq \log_2 |S| - c$), and not that S also lead to a near-optimal two-part program for x (i.e., that $K(S) + \log_2 |S| \leq K(x) + c$). More formally, Mota et al. define the *naïve c-sophistication* of x, or $\operatorname{nsoph}_c(x)$, to be the minimum of $K(S)$ over all models S for x such that $K(x|S) \geq \log_2 |S| - c$. By point (iii) above, it is clear that $\operatorname{nsoph}_c(x) \leq \operatorname{soph}_c(x)$. *A priori*, $\operatorname{nsoph}_c(x)$ could be much smaller $\operatorname{soph}_c(x)$, thereby leading to two different sophistication notions. However, it follows from an important 2004 result of Vereshchagin and Vitányi [10] that $\operatorname{soph}_{c+O(\log n)}(x) \leq \operatorname{nsoph}_c(x)$ for all x, and hence the two notions are basically equivalent.

Sophistication is sometimes criticized for being "brittle": it is known that increasing the parameter c only slightly can cause $\operatorname{soph}_c(x)$ and $\operatorname{nsoph}_c(x)$ to fall drastically, say from $n - O(\log n)$ to $O(1)$. However, a simple fix to that problem is to consider the quantities $\min_c \{c + \operatorname{soph}_c(x)\}$ and $\min_c \{c + \operatorname{nsoph}_c(x)\}$. Those are known, respectively, as the *coarse sophistication* $\operatorname{csoph}(x)$ and *naïve coarse sophistication* $\operatorname{ncsoph}(x)$, and they satisfy $\operatorname{ncsoph}(x) \leq \operatorname{csoph}(x) \leq \operatorname{ncsoph}(x) + O(\log n)$.

The advantage of sophistication is that it captures, more cleanly than any other measure, what exactly we mean by "interesting" versus "random" information. Unlike with coarse entropy, with sophistication there's no need to specify a denoising function f, with all the risk of arbitrariness that entails. Instead, if one likes, the definition of sophistication picks out a denoising function for us: namely, whatever function maps x to its corresponding model S.

Unfortunately, this conceptual benefit comes at a huge computational price. Just as $K(x)$ is uncomputable, so one can show that the sophistication measures are uncomputable as well. But with $K(x)$, at least we can get better and better upper bounds, by finding smaller and smaller compressed representations for x. By contrast, even to *approximate* sophistication requires solving two coupled optimization problems: firstly over possible models S, and secondly over possible ways to specify x given S.

A second disadvantage of sophistication is that, while there *are* highly-sophisticated strings, the only known way to produce such a string (even probabilistically) is via a somewhat-exotic diagonalization argument. (By contrast, for "reasonable" choices of denoising function f, one can easily generate x for which the coarse entropy $H(f(x))$ is large.) Furthermore, this is not an accident, but an unavoidable consequence of sophistication's generality. To see this, consider any short probabilistic program P: for example, the coffee automaton that we will study in this paper, which has a simple initial state and a simple probabilistic evolution rule. Then we claim that *with overwhelming probability, P's output x must have low sophistication.* For as the model S, one can take the set of all possible outputs y of P such that $\Pr[y] \approx \Pr[x]$. This S takes only $O(\log n)$ bits to describe (plus $O(1)$ bits for P itself), and clearly $K(x|S) \geq \log_2 |S| - c$ with high probability over x.

For this reason, sophistication as defined above seems irrelevant to the coffee cup or other physical systems: it simply never becomes large for such systems! On the other hand, note that the two drawbacks of sophistication might "cancel each other out" if we consider *resource-bounded* versions of sophistication: that is, versions where we impose constraints (possibly severe constraints) on both the program for generating S, and the program for generating x given S. Not only does the above argument fail for resource-bounded versions of sophistication, but those versions are the only ones we can hope to compute anyway! With Kolmogorov complexity, we're forced to consider proxies (such as `gzip` file size) mostly just because $K(x)$ itself is uncomputable. By contrast, even if we *could* compute $\operatorname{soph}_c(x)$ perfectly, it would never become large for the systems that interest us here.

2.3 Logical Depth

A third notion, introduced by Bennett [2], is *logical depth.* Roughly speaking, the logical depth of a string x is the amount of time taken by the shortest program that outputs x. (Actually, to avoid the problem of "brittleness," one typically considers something like the minimum amount of time taken by any program that outputs x and whose length is at most $K(x) + c$, for some constant "fudge factor" c. This is closely analogous to what is done for sophistication.)

The basic idea here is that, both for simple strings and for random ones, the shortest program will *also* probably run in nearly linear time. By contrast, one can show that there exist “deep” strings, which can be generated by short programs but only after large amounts of time.

Like sophistication, logical depth tries to probe the internal structure of a minimal program for x—and in particular, to distinguish between the “interesting code” in that program and the “boring data” on which the code acts. The difference is that, rather than trying to measure the *size* of the “interesting code,” one examines how long it takes to run.

Bennett [2] has advocated logical depth as a complexity measure, on the grounds that logical depth encodes the “amount of computational effort” used to produce x, according to the “most probable” (i.e., lowest Kolmogorov complexity) hypothesis about how x was generated. On the other hand, an obvious disadvantage of logical depth is that it’s even less clear how to estimate it in practice than was the case for sophistication.

A second objection to logical depth is that even short, fast programs can be extremely “complicated” in their behavior (as evidenced, for example, by cellular automata such as Conway’s Game of Life). Generating what many people would regard as a visually complex pattern—and what *we* would regard as a complex, milk-tendril-filled state, in the coffee-cup system—simply need not take a long time! For this reason, one might be uneasy with the use of running time as a proxy for complexity.

2.4 Light-Cone Complexity

The final complexity measure we consider was proposed by Shalizi, Shalizi, and Haslinger [9]; we call it *light-cone complexity.* In contrast to the previous measures, light-cone complexity does not even try to define the “complexity” of a string x, given only x itself. Instead, the definition of light-cone complexity assumes a *causal structure*: that is, a collection of spacetime points $\mathcal{A}$ (assumed to be fixed), together with a transitive, cycle-free binary relation indicating which points $a \in \mathcal{A}$ are to the “future” of which other points in $\mathcal{A}$. The set of all points $b \in \mathcal{A}$ to a’s future is called a’s *future light-cone*, and is denoted $F(a)$. The set of all points $b \in \mathcal{A}$ to a’s past (that is, such that a is to b’s future) is called a’s *past light-cone*, and is denoted $P(a)$. For example, if we were studying the evolution of a 1-dimensional cellular automaton, then $\mathcal{A}$ would consist of all ordered pairs (x,t) (where x is position and t is time), and we would have

$$\begin{aligned} F(x,t) &= \{(y,u) : u > t, \quad |x-y| \leq u-t\}, \\ P(x,t) &= \{(y,u) : u < t, \quad |x-y| \leq t-u\}. \end{aligned}$$

Now given a spacetime point $a \in \mathcal{A}$, let V_a be the actual value assumed by the finite automaton at a (for example, “alive” or “dead,” were we discussing Conway’s Game of Life or some other 2-state system). In general, the finite automaton might be probabilistic, in which case V_a is a random variable, with a Shannon entropy $H(V_a)$ and so forth. Also, given a set $S \subseteq \mathcal{A}$, let $V_S := (V_a)_{a\in S}$ be a complete description of the values at *all* points in S. Then the light-cone complexity at a point $a \in \mathcal{A}$, or $\mathrm{LCC}(a)$, can be defined as follows:

$$\begin{aligned} \mathrm{LCC}(a) &= I\left(V_{P(a)} : V_{F(a)}\right) \\ &= H\left(V_{P(a)}\right) + H\left(V_{F(a)}\right) - H\left(V_{P(a)}, V_{F(a)}\right). \end{aligned}$$

In other words, $\mathrm{LCC}(a)$ is the *mutual information* between a’s past and future light-cones: the number of bits about a’s future that are encoded by its past. If we want the light-cone complexity of (say) an entire spatial slice, we could then take the sum of $\mathrm{LCC}(a)$ over all a in that slice, or some other combination.

The intuition here is that, if the cellular automaton dynamics are "too simple," then $\mathrm{LCC}(a)$ will be small simply because $H\left(V_{P(a)}\right)$ and $H\left(V_{F(a)}\right)$ are both small. Conversely, if the dynamics are "too random," then $\mathrm{LCC}(a)$ will be small because $H\left(V_{P(a)}, V_{F(a)}\right) \approx H\left(V_{P(a)}\right) + H\left(V_{F(a)}\right)$: although the past and future light-cones both have plenty of entropy, they are uncorrelated, so that knowledge of the past is of barely any use in predicting the future. Only in an intermediate regime, where there are interesting *non*-random dynamics, should there be substantial uncertainty about $V_{F(a)}$ that can be reduced by knowing $V_{P(a)}$.

As Shalizi et al. [9] point out, a major advantage of light-cone complexity, compared to sophistication, logical depth, and so on, is that light-cone complexity has a clear "operational meaning": it is easy to state the question that light-cone complexity is answering. That question is the following: "how much could I possibly predict about the configurations in a's future, given complete information about a's past?" The reason to focus on light-cones, rather than other sets of points, is that the light-cones are automatically determined once we know the causal structure: there seems to be little arbitrariness about them.

On the other hand, depending on the application, an obvious drawback of light-cone complexity is that it can't tell us the "inherent" complexity of an object x, without knowing about x's past and future. If we wanted to use a complexity measure to make *inferences* about x's past and future, this might be seen as question-begging. A less obvious drawback arises if we consider a dynamical system that changes slowly with time: for example, a version of the coffee automaton where just a single cream particle is randomly moved at each time step. Consider such a system in its "late" stages: that is, after the coffee and cream have fully mixed. Even then, Shalizi et al.'s $\mathrm{LCC}(a)$ measure will remain large, but not for any "interesting" reason: only because a's past light-cone will contain almost the same (random) information as its future light-cone, out to a very large distance! Thus, LCC seems to give an intuitively wrong answer in these cases (though no doubt one could address the problem by redefining LCC in some suitable way).

The computational situation for LCC seems neither better nor worse to us than that for (say) coarse entropy or resource-bounded sophistication. Since the light-cones $P(a)$ and $V(a)$ are formally infinite, a first step in estimating $\mathrm{LCC}(a)$—as Shalizi et al. point out—is to impose some finite cutoff t on the number of steps into a's past and future one is willing to look. Even then, one needs to estimate the mutual information $I\left(V_{P_t(a)} : V_{F_t(a)}\right)$ between the truncated light-cones $P_t(a)$ and $F_t(a)$, a problem that naïvely requires a number of samples exponential in t. One could address this problem by simply taking t extremely small (Shalizi et al. set $t = 1$). Alternatively, if a large t was needed, one could use the same Kolmogorov-complexity-based approach that we adopt in this paper for coarse entropy. That is, one first replaces the mutual information by the mutual *algorithmic* information

$$K\left(V_{P_t(a)} : V_{F_t(a)}\right) = K\left(V_{P_t(a)}\right) + K\left(V_{F_t(a)}\right) - K\left(V_{P_t(a)}, V_{F_t(a)}\right),$$

and then estimates $K(x)$ using some computable proxy such as `gzip` file size.

2.5 Synthesis

It seems like we have a bestiary of different complexity notions. Fortunately, the four notions discussed above can all be related to each other; let us discuss how.

First, one can view coarse entropy as a kind of "resource-bounded" sophistication. To see this, let f be any coarse-graining function. Then $K(f(x))$, the Kolmogorov complexity of $f(x)$, is essentially equal to $K(S_{f,x})$, where

$$S_{f,x} := \{y : f(y) = f(x)\}. \tag{1}$$

Thus, if instead of minimizing over *all* models S for x that satisfy some condition, we consider only the particular model $S_{f,x}$ above, then sophistication reduces to coarse entropy. Note that this argument establishes neither that coarse entropy is an upper bound on sophistication, nor that it's a lower bound. Coarse entropy could be larger, if the minimization found some model S for x with $K(S) \ll K(S_{f,x})$. But conversely, sophistication could also be larger, if the model $S_{f,x}$ happened to satisfy $K(x|S_{f,x}) \ll \log_2 |S_{f,x}|$ (that is, x was a highly "non-generic" element of $S_{f,x}$).

Second, Antunes and Fortnow [1] proved a close relation between coarse sophistication and a version of logical depth. Specifically, the Busy Beaver function, $\mathrm{BB}(k)$, is defined as the maximum number of steps for which a k-bit program can run before halting when given a blank input. Then given a string x, Antunes and Fortnow [1] define the *Busy Beaver computational depth* $\mathrm{depth}_{BB}(x)$ to be the minimum, over all programs p that output a model S for x in $BB(k)$ steps or fewer, of $|p| + k - K(x)$. They then prove the striking result that csoph and depth_{BB} are essentially equivalent: for all $x \in \{0,1\}^n$,

$$|\mathrm{csoph}(x) - \mathrm{depth}_{BB}(x)| = O(\log n).$$

Third, while light-cone complexity is rather different from the other three measures (due to its taking as input an entire causal history), it can be loosely related to coarse entropy as follows. If $\mathrm{LCC}(a)$ is large, then the region around a must contain large "contingent structures": structures that are useful for predicting future evolution, but that might have been different in a different run of the automaton. And one might expect those structures to lead to a large coarse entropy in a's vicinity. Conversely, if the coarse entropy is large, then one expects contingent structures (such as milk tendrils, in the coffee automaton), which could then lead to nontrivial mutual information between a's past and future light-cones.

Having described four complexity measures, their advantages and disadvantages, and their relationships to each other, we now face the question of which measure to use for our experiment. While it would be interesting to study the rise and fall of light-cone complexity in future work, here we decided to restrict ourselves to complexity measures that are functions of the current state. That leaves coarse entropy, sophistication, and logical depth (and various approximations, resource-bounded versions, and hybrids thereof).

Ultimately, we decided on a type of coarse entropy. Our reason was simple: because even after allowing resource bounds, *we did not know of any efficient way to approximate sophistication or logical depth.* In more detail, given a bitmap image x of a coffee cup, our approach first "smears x out" using a denoising function f, then uses the `gzip` file size of $f(x)$ as an upper bound on the Kolmogorov complexity $K(f(x))$ (which, in turn, is a proxy for the Shannon entropy $H(f(x))$ of $f(x)$ considered as a random variable). There are a few technical problems that arise when implementing this approach (notably, the problem of "border pixel artifacts"). We discuss those problems and our solutions to them in Section 4.

Happily, as discussed earlier in this section, our coarse entropy measure can be related to the other measures. For example, coarse entropy can be seen as an *extremely* resource-bounded variant of sophistication, with the set $S_{f,x}$ of equation (1) playing the role of the model S. As discussed in Section 2.1, one might object to our coarse entropy measure on the grounds that our denoising function f is "arbitrary," that we had no principled reason to choose it rather than some other function. Interestingly, though, one can answer that objection by taking inspiration from light-cone complexity. Our denoising function f will *not* be completely arbitrary, for the simple reason that the regions over which we coarse-grain—namely, squares of contiguous cells—will correspond to the coffee automaton's causal structure.[3]

[3] Technically, if we wanted to follow the causal structure, then we should have used *diamonds* of continguous cells

3 The Coffee Automaton

The coffee cup system that we use as our model is a simple stochastic cellular automaton. A two-dimensional array of bits describes the system's state, with ones representing particles of cream, and zeros representing particles of coffee. The cellular automaton implementation used for this project is written in Python; source code is available for download.[4]

The automaton begins in a state in which the top half of the cells are filled with ones, and the bottom half is filled with zeros. At each time step, the values in the cells change according to a particular transition rule. We consider two different models of the coffee cup system, each having its own transition rule.

3.1 Interacting Model

In the interacting model of the coffee cup system, only one particle may occupy each cell in the state array. The transition rule for this model is as follows: at each time step, one pair of horizontally or vertically adjacent, differing particles is selected, and the particles' positions are swapped. This model is *interacting* in the sense that the presence of a particle in a cell prevents another particle from entering that cell. The movements of particles in this model are not independent of one another.

This model reflects the physical principle that two pieces of matter may not occupy the same space at the same time. However, the interactions between particles that make this model more realistic also make it harder to reason about theoretically.

3.2 Non-Interacting Model

In the non-interacting model of the coffee cup system, any number of cream particles may occupy a single cell in the state array. Coffee particles are not considered important in this model; they are simply considered a background through which the cream particles move. The transition rule for this model is as follows: at each time step, each cream particle in the system moves one step in a randomly chosen direction. This model is *non-interacting* in that the location of each cream particle is independent of all the others. The presence of a cream particle in a particular cell does not prevent another cream particle from also moving into that cell.

We consider this model because it is easier to understand theoretically. Since the particles in the system do not interact, each particle can be considered to be taking an independent random walk. The dynamics of random walks are well-understood, so it is easy to make theoretical predictions about this model (see Appendix 9) and compare them to the experimental results.

4 Approximating Coarse Entropy

While Kolmogorov complexity and sophistication are useful theoretical notions to model our ideas of entropy and complexity, they cannot be directly applied in numerical simulations, because they are both uncomputable. As such, while we use these concepts as a theoretical foundation, we need to develop algorithms that attempt to approximate them.

Evans et al. [3] propose an algorithm, called the optimal symbol compression ratio (OSCR) algorithm, which directly estimates Kolmogorov complexity and sophistication. Given an input

rather than squares. But this difference is presumably insignificant.

[4] At `www.scottaaronson.com/coffee_automaton.zip`

string x, the OSCR algorithm produces a two-part code. The first part is a codebook, which maps symbols chosen from the original input string to new symbols in the encoded string. The second part of the code is the input string, encoded using the symbols in this codebook. The goal of OSCR is to select which symbols to put in the codebook such that the total size of the output—codebook size plus encoded string size—is minimized. The optimal codebook size for x is an estimate of $K(S)$, the sophistication of x. The optimal total size of the output for x is called the minimum description length (MDL) of the string, and is an estimate of $K(x)$.

The OSCR approach seems promising because of its direct relationship to the functions we are interested in approximating. However, we implemented a version of this algorithm, and we found that our implementation does not perform well in compressing the automaton data. The output of the algorithm is noisy, and there is no obvious trend in either the entropy or complexity estimates. We conjecture that the noise is present because this compression method, unlike others we consider, does not take into account the two-dimensionality of the automaton state.

An alternative metric adopts an idea from physics called *coarse-graining*. Coarse-graining aims to describe a system's state on a macroscopic scale—for example, the coffee cup as it would be seen by a human observer from a few feet away—by essentially "blurring" the state, averaging nearby values together. In contrast, the original, or fine-grained, version of the coffee cup's state—the complete array of bits representing the individual particles in the cup—describes the system on a microscopic scale.

Conceptually, for an automaton state represented by a string x, its coarse-grained version is analogous to a typical set S which contains x. The coarse-grained state describes the high-level, "non-random" features of x—features which it has in common with all other states from which the same coarse-grained representation could be derived. Thus, the descriptive size of the coarse-grained state can be used as an estimate for the state's sophistication, $K(S)$. To estimate the descriptive size of the coarse-grained state, we compress it using a general file compression program, such as `gzip` or `bzip`. Shalizi [8] objects to the use of such compression programs, claiming that they do not provide consistently accurate entropy estimates and that they are too slow. In our experiments, we have not seen either of these problems; our simulations run in a reasonable amount of time and produce quite consistent entropy estimates (see, for instance, Figure 5). We therefore use such compression programs throughout, though we consider alternative approaches in Section 7.

Having defined the notion of coarse-graining, we can then define a two-part code based on it. If the first part of the code—the typical set—is the coarse-grained state, then the second part is $K(x|S)$, the information needed to reconstruct the fine-grained state given the coarse-grained version. The total compressed size of both parts of the code is an estimate of the Kolmogorov complexity of the state, $K(x)$.

We attempted to implement such a two-part code, in which the second part was a diff between the fine-grained and coarse-grained states. The fine-grained state, x, could be uniquely reconstructed from the coarse-grained array and the diff. In our implementation of this two-part code, our estimate of $K(x|S)$ suffered from artifacts due to the way the diff was represented. However, defining a two-part code based on coarse-graining is possible in general.

In light of the artifacts produced by our implementation of the two-part code, we chose to pursue a more direct approach using coarse-graining. We continued to use the compressed size of the coarse-grained state as an approximation of $K(S)$. However, instead of approximating $K(x|S)$ and using $K(S) + K(x|S)$ as an estimate of $K(x)$, we approximated $K(x)$ directly, by measuring the compressed size of the fine-grained array. This approach avoided the artifacts of the diff-based code, and was used to generate the results reported here.

5 Coarse-Graining Experiment

5.1 Method

To derive a coarse-grained version of the automaton state from its original, fine-grained version, we construct a new array in which the value of each cell is the average of the values of the nearby cells in the fine-grained array. We define "nearby" cells as those within a $g \times g$ square centered at the cell in question. The value of g is called the grain size, and here is selected experimentally. This procedure is illustrated in Figure 1.

Fine-Grained

0	1	0	0	1
0	1	1	0	1
0	1	0	1	0
1	0	1	1	1
1	0	1	0	0

Coarse-Grained

0.5	0.5	0.5	0.5	0.5
0.5	$0.\overline{4}$	$0.\overline{4}$	$0.\overline{4}$	0.5
0.5	$0.\overline{5}$	$0.\overline{6}$	$0.\overline{6}$	$0.\overline{3}$
0.5	$0.\overline{5}$	$0.\overline{5}$	$0.\overline{5}$	0.5
0.5	$0.\overline{6}$	0.5	$0.\overline{6}$	0.5

Figure 1: Illustration of the construction of the coarse-grained array, using an example grain size of 3. The values of the shaded cells at left are averaged to produce the value of the shaded cell at right.

Given this array of averages, we then threshold its floating-point values into three buckets. Visually, these buckets represent areas which are mostly coffee (values close to 0), mostly cream (values close to 1), or mixed (values close to 0.5). The estimated complexity of the state, $K(S)$, is the file size of the thresholded, coarse-grained array after compression. Analogously, the estimated entropy of the automaton state is the compressed file size of the fine-grained array.

5.2 Results and Analysis

Results from simulation of the automaton using the coarse-graining metric are shown in Figure 2.

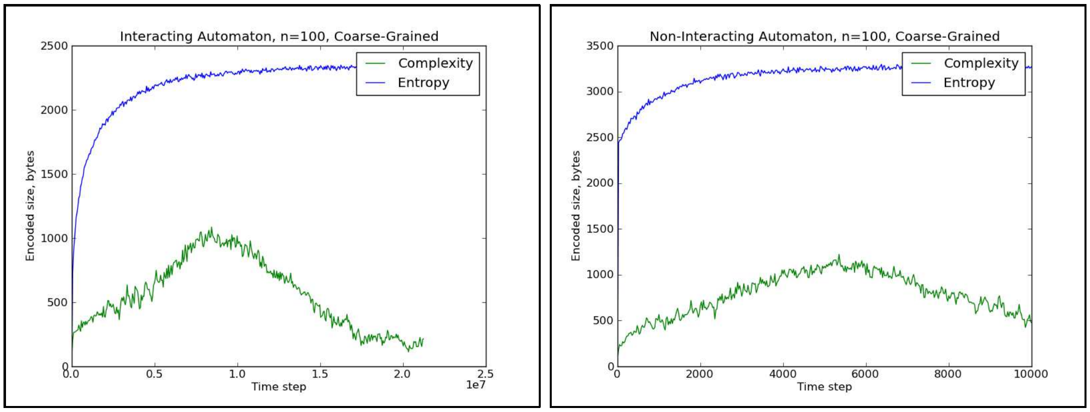

Figure 2: The estimated entropy and complexity of an automaton using the coarse-graining metric. Results for the interacting model are shown at left, and results for the non-interacting model are at right.

Both the interacting and non-interacting models show the predicted increasing, then decreasing pattern of complexity. Both models also have an increasing entropy pattern, which is expected due to the second law of thermodynamics. The initial spike in entropy for the non-interacting automaton can be explained by the fact that all of the particles can move simultaneously after the first time step. Thus, the number of bits needed to represent the state of the non-interacting automaton jumps after the first time step. With the interacting automaton, by contrast, particles far from the coffee-cream border cannot move until particles closer to the border have moved, so there is less change in the automaton at each time step. Therefore, the estimated entropy of this model is predictably more continuous throughout.

A visualization of the automaton's changing state over time is provided in Figures 3 and 4. This visualization is generated by converting each cell's value to a grayscale color value; lighter colors correspond to larger values. Visually, the fine-grained representation of the state continues to grow more complicated with time, while the coarse-grained representation first becomes first more and then less complicated.

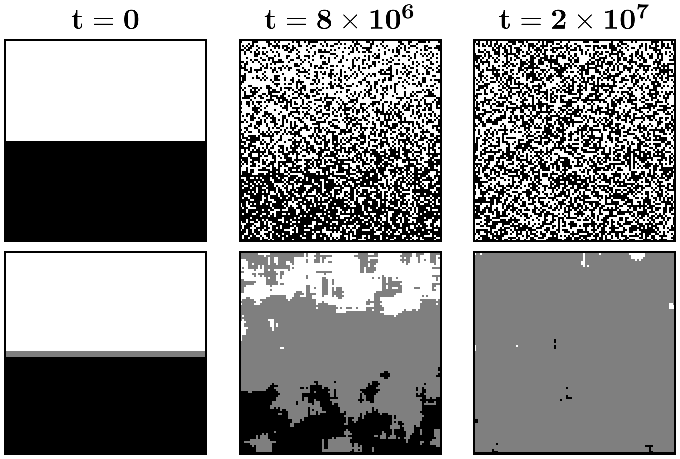

Figure 3: Visualization of the state of the interacting automaton of size 100 over time. The top row of images is the fine-grained array, used to estimate entropy. The bottom row is the coarse-grained array, used to estimate complexity. From left to right, the images represent the automaton state at the beginning of the simulation, at the complexity maximum, and at the end of the simulation.

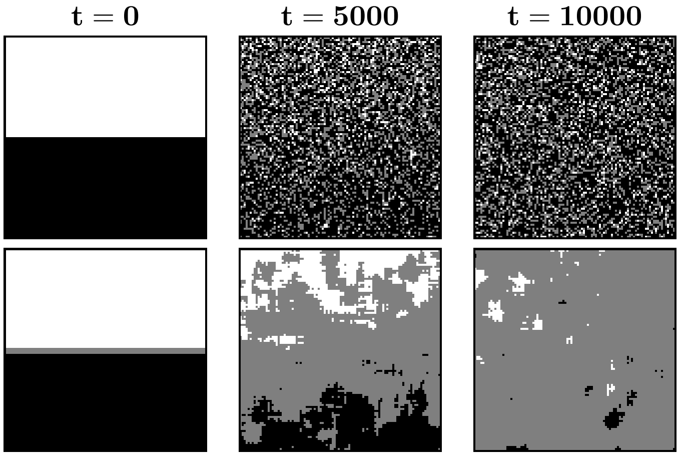

Figure 4: Visualization of the state of the non-interacting automaton of size 100 over time.

The gzip compression algorithm was used to generate the results in Figure 2, and is used throughout when a general file compression program is needed. The results achieved using the coarse-graining metric are qualitatively similar when different compression programs are used, as shown in Figure 5.

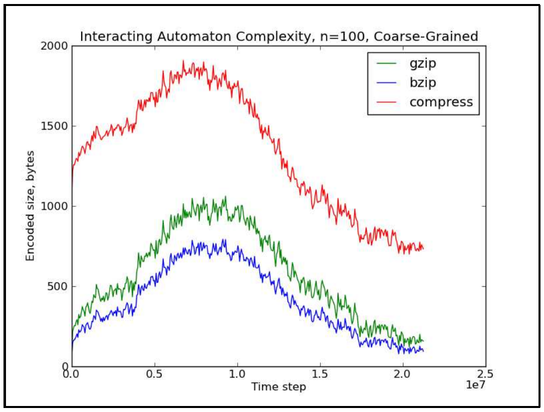

Figure 5: Coarse-grained complexity estimates for a single simulation of the interacting automaton, using multiple file compression programs.

Given these results, it is informative to examine how complexity varies with n, the size of the automaton.

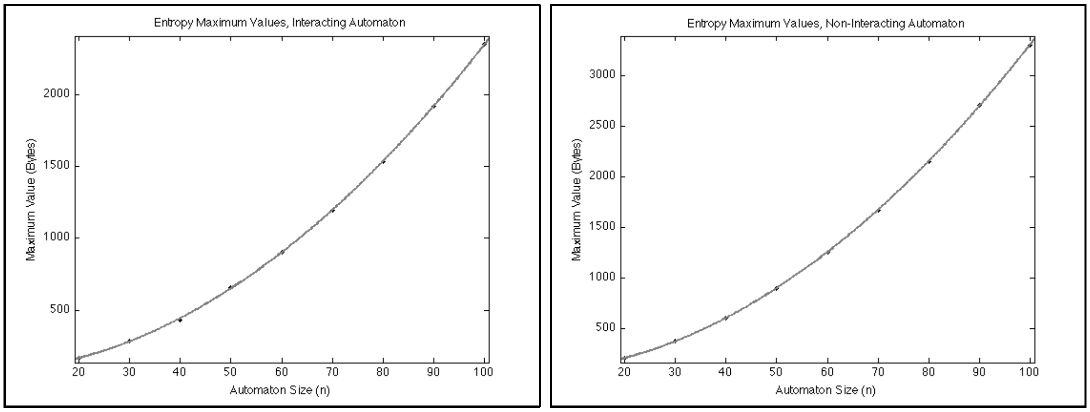

Figure 6: Graphs of automaton size versus entropy maximum value. Quadratic curve fits are shown, with r^2 values of 0.9999 for both the interacting and non-interacting automaton.

The well-fit quadratic curve for the maximum values of entropy (Figure 6) is expected. The maximum entropy of an automaton is proportional to the number of particles in the automaton. This is because, if the state of the automaton is completely random, then the compressed size of the state is equal to the uncompressed size–the number of particles. As the automaton size, n, increases, the number of particles increases to n^2.

The maximum values of complexity appear to increase linearly as the automaton size increases (Figure 7). That is, maximum complexity is proportional to the side length of the two-dimensional state array. This result is expected, since the automaton begins in a state which is symmetric along its vertical axis, and complexity presumably develops along a single dimension of the automaton. The time that it takes for the automaton to reach its complexity maximum appears to increase quadratically with the automaton size, or proportionally to the number of particles in the automaton

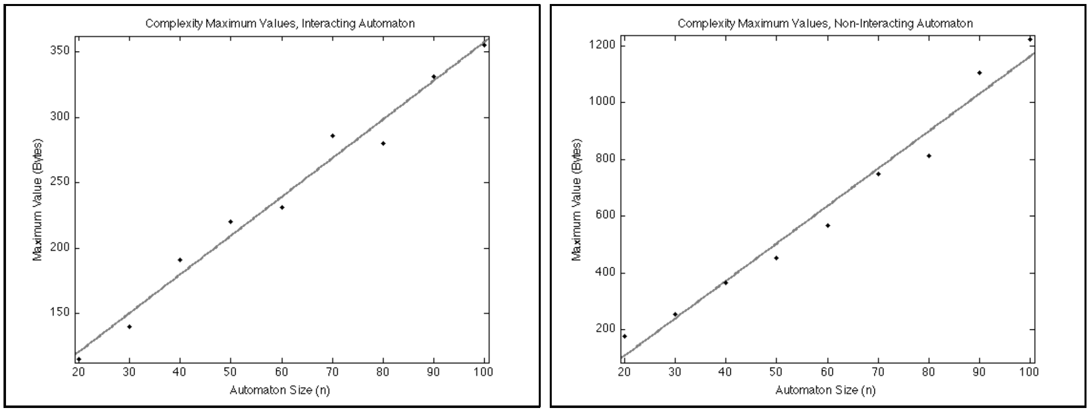

Figure 7: Graphs of automaton size versus complexity maximum value. Linear curve fits are shown, with r^2 values of 0.9798 for the interacting automaton and 0.9729 for the non-interacting automaton.

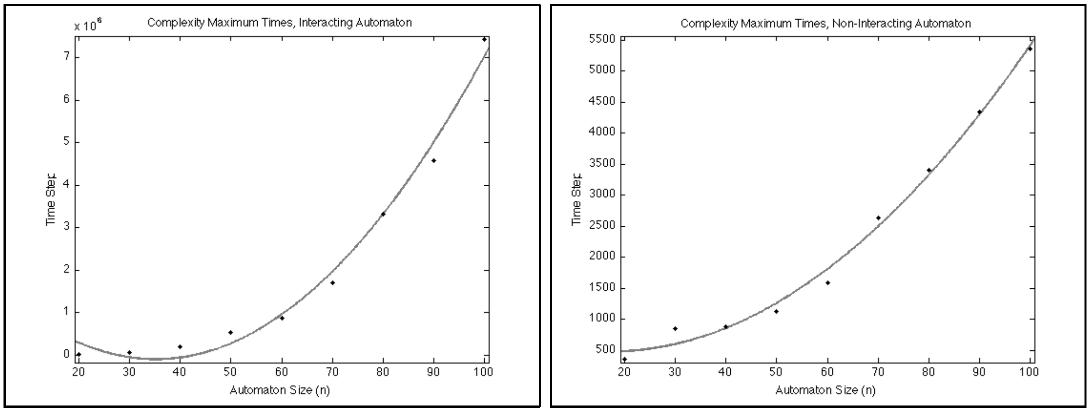

Figure 8: Graphs of automaton size versus time to complexity maximum. Quadratic curve fits are shown, with r^2 values of 0.9878 for the interacting automaton and 0.9927 for the non-interacting automaton.

(Figure 8). This result is also expected, since the time for n^2 particles to reach a particular configuration is proportional to n^2.

6 Adujsted Coarse-Graining Experiment

6.1 Method

Though the original coarse-graining approach produces the hypothesized complexity pattern, the method of thresholding used in the previous experiment—dividing the floating-point values into three buckets—has the potential to introduce artificial complexity. Consider, for example, an automaton state for which the coarse-grained array is a smooth gradient from 0 to 1. By definition, there will be some row of the array which lies on the border between two threshold values. Tiny fluctuations in the values of the coarse-grained array may cause the cells in this row to fluctuate between two threshold values. In such a case, the small fluctuations in this border row would artificially increase the measured complexity of the coarse-grained array. This case is illustrated in Figure 9.

We propose an adjustment to the coarse-graining algorithm that helps to minimize these artifacts. First, we use a larger number of thresholds—seven, in contrast to the three used in the

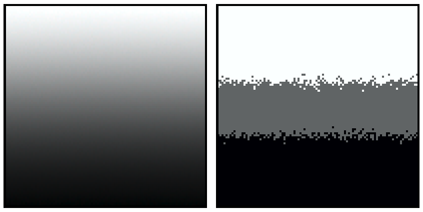

Figure 9: A coarse-grained array consisting of a smooth gradient from 0 to 1 is shown at left. At right is the same array after a small amount of simulated noise has been added and the values have been thresholded.

original experiment. Additionally, we allow each cell in the array to be optionally, independently adjusted up or down by one threshold, in whatever manner achieves the smallest possible file size for the coarse-grained array.

This adjustment helps to compensate for the thresholding artifacts–such random fluctuations could be removed by adjusting the fluctuating pixels. However, since each pixel can be adjusted independently, there are 2^{n^2} possible ways to adjust a given coarse-grained array.

Because we cannot search through this exponential number of possible adjustments to find the optimal one, we develop an approximation algorithm to produce an adjustment that specifically targets pixels on the border between two thresholds. Given the properties of the automaton—it begins with rows of dark cells on top, and light cells on the bottom—it is likely that each row of the coarse-grained array will contain similar values. Thus, we adjust the coarse-grained array by using a majority algorithm. If a cell is within one threshold value of the majority value in its row, it is adjusted to the majority value.

The hope is that

(1) this adjustment will reduce artificial border complexity by "flattening" fluctuating border rows to a single color,

(2) the adjustment will not eliminate actual complexity, since complicated structures will create value differences in the coarse-grained array that are large enough to span multiple thresholds.

6.2 Results and Analysis

Results from simulation of the automaton using the adjusted coarse-graining metric are shown in Figure 10. Visualizations of the automaton state are shown in Figures 11 and 12.

While this metric is somewhat noisier than the original coarse-graining method, it results in a similarly-shaped complexity curve for the interacting automaton. For the non-interacting automaton, however, the complexity curve is flattened to a lower value.

This result for the non-interacting automaton is actually borne out by theoretical predictions. The basic story is as follows; for details, see Appendix 9. If we consider the automaton state to "wrap around" from right to left, then by symmetry, the expected number of cream particles at a particular location in the automaton depends solely on the vertical position of that location. The expectations of all cells in a particular row will be the same, allowing the two-dimensional automaton state to be specified using a single dimension. Modeling each particle of cream as taking a random walk from its initial position, it is possible to calculate the expected number of particles at a given position as a function of time. Further, Chernoff bounds can be used to demonstrate that the actual number of particles in each grain of the coarse-grained state is likely to be close to the expectation, provided that the grain size is large enough. Since it is possible to

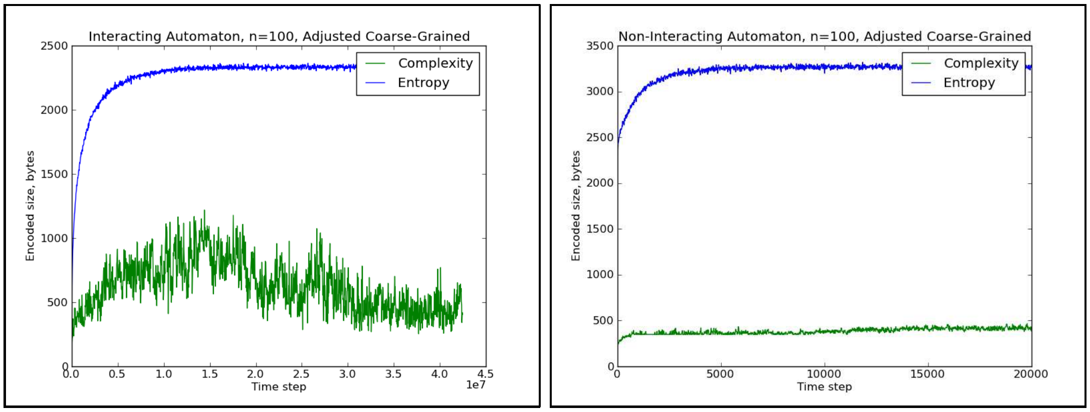

Figure 10: The estimated entropy and complexity of an automaton using the adjusted coarse-graining metric.

specify the expected distribution of particles in the non-interacting automaton at all times using such a function, the complexity of the non-interacting automaton state is always low.

We believe thresholding artifacts caused the apparent increase in complexity for the non-interacting automaton when regular coarse-graining was used. Our adjustment removes all of this estimated complexity from the non-interacting automaton, but preserves it in the interacting automaton. This evidence suggests that the interacting automaton model may actually have intermediate states of high complexity, even if the non-interacting model never becomes complex.

7 Conclusions and Further Work

Of the metrics considered in this project, the coarse-graining approaches provide the most effective estimate of complexity that produces results which mirror human intuition. However, this metric is flawed specifically *because* it is based on human intuition and perceptions of complexity. Ideally, a complexity metric would be found which produces similar results without relying on such assumptions. The OSCR approach seems promising for its independence from these assumptions and for its theoretical foundations. It is possible that a different implementation of this algorithm could produce better results than the one we used for this project.

It would also be worthwhile to investigate other complexity metrics, beyond those already explored in this paper. Shalizi et al. [9] propose a metric based on the concept of light cones. They define $C(x)$, the complexity of a point x in the spacetime history, as the mutual information between descriptions of its past and future light cones. Letting $P(x)$ be the past light cone and $F(x)$ the future light cone, $C(x) = H(P(x)) + H(F(x)) - H(P(x), F(x))$. This metric is of particular interest because it avoids the problem of artifacts created by coarse-graining; it can also be approximated in a way that avoids the use of `gzip`. Running experiments with the automaton using the light cone metric, and comparing the results to those generated using coarse-graining, could provide more information about both metrics.

Ultimately, numerical simulation is of limited use in reasoning about the problem of complexity. Approximation algorithms can provide only an upper bound, not a lower bound, on Kolmogorov complexity and sophistication. To show that a system really does become complex at intermediate points in time, it is necessary to find a lower bound for the system's complexity. Future theoretical work could help provide such a lower bound, and could also generate further insight into the origins of complexity in closed systems.

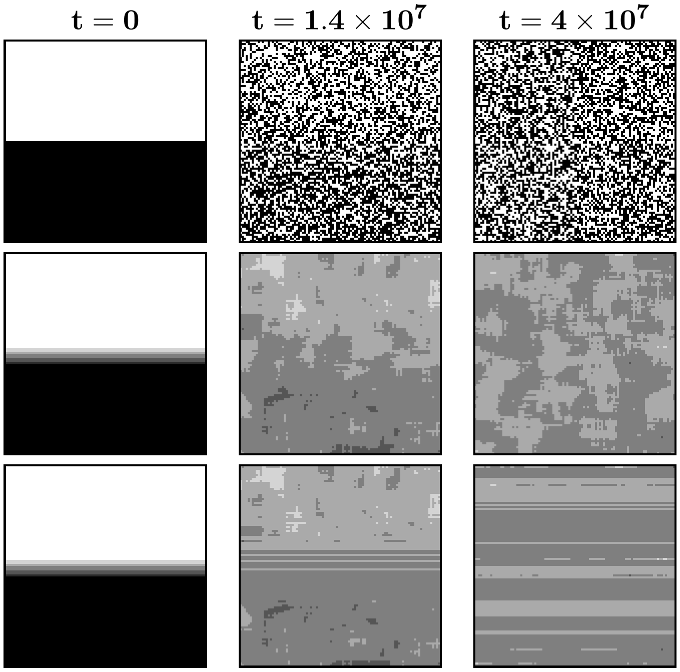

Figure 11: Visualization of the state of the interacting automaton of size 100 over time. The rows of images represent the fine-grained state, the original coarse-grained state, and the coarse-grained state after adjustment, respectively.

8 Acknowledgments

We thank Alex Arkhipov, Charles Bennett, Ian Durham, Dietrich Leibfried, Aldo Pacchiano, and Luca Trevisan for helpful discussions.

References

[1] L. Antunes and L. Fortnow. Sophistication revisited. *Theory of Computing Systems*, 45(1):150–161, 2009.

[2] C. H. Bennett. Logical depth and physical complexity. In *The Universal Turing Machine A Half-Century Survey*, pages 207–235. Springer, 1995.

[3] S. Evans, G. Saulnier, and S. Bush. A new universal two part code for estimation of string Kolmogorov complexity and algorithmic minimum sufficient statistic. In *DIMACS Workshop on Complexity and Inference*, 2003. `http://www.stat.ucla.edu/~cocteau/dimacs/evans.pdf`.

[4] P. Gács, J. Tromp, and P. M. B. Vitányi. Algorithmic statistics. *IEEE Trans. Information Theory*, 47(6):2443–2463, 2001.

[5] M. Koppel. Complexity, depth, and sophistication. *Complex Systems*, 1(6):1087–1091, 1987.

[6] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications (1st edition).* Springer-Verlag, 1993.

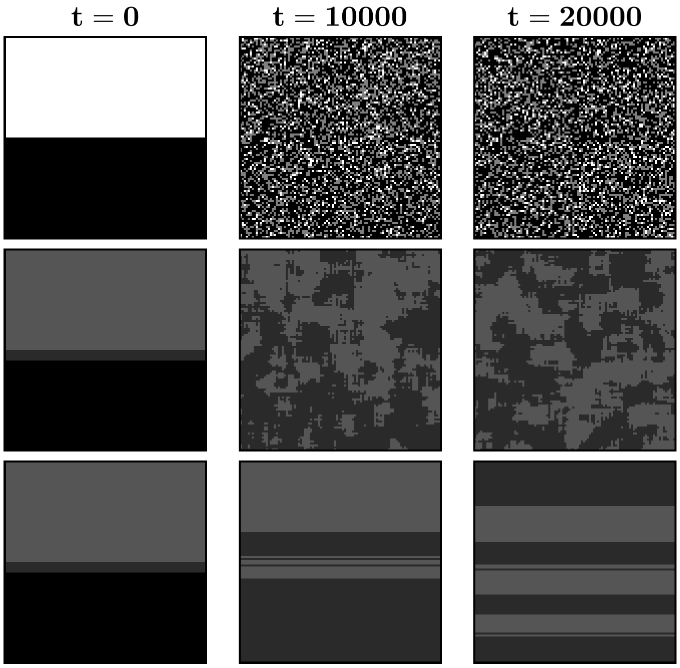

Figure 12: Visualization of the state of the non-interacting automaton of size 100 over time. Note that the coarse-grained images are darker than for the previous coarse-graining metric, because a larger number of thresholds were used.

[7] F. Mota, S. Aaronson, L. Antunes, and A. Souto. Sophistication as randomness deficiency. In *Descriptional Complexity of Formal Systems*, volume 8031 of *Lecture Notes in Computer Science*, pages 172–181, 2013.

[8] C. R. Shalizi. Methods and techniques of complex systems science: an overview. In T. S. Deisboeck and J. Y. Kresh, editors, *Complex Systems Science in Biomedicine*, pages 33–114. Springer, 2006. nlin.AO/0307015.

[9] C. R. Shalizi, K. L. Shalizi, and R. Haslinger. Quantifying self-organization with optimal predictors. *Physical Review Letters*, 93(118701), 2004. nlin.AO/0409024.

[10] N. Vereshchagin and P. Vitányi. Kolmogorov's structure functions with an application to the foundations of model selection. In *IEEE Foundations of Computer Science (FOCS)*, pages 751–760, 2002.

9 Appendix: The Non-Interacting Case

Let's consider the non-interacting coffee automaton on an $n \times n$ grid with periodic boundary conditions. At each time step, each cream particle moves to one of the 4 neighboring pixels uniformly at random. Let $a_t(x, y)$ be the number of cream particles at point (x, y) after t steps.

Claim 1. *For all x, y, t, we have* $\mathrm{E}[a_t(x, y)] \leq 1$.

Proof. By induction on t. If $t = 0$, then $a_0(x,y) \in \{0,1\}$. Furthermore, by linearity of expectation,

$$\mathrm{E}[a_{t+1}(x,y)] = \frac{\mathrm{E}[a_t(x-1,y)] + \mathrm{E}[a_t(x+1,y)] + \mathrm{E}[a_t(x,y-1)] + \mathrm{E}[a_t(x,y+1)]}{4}.$$

□

Now let B be an $L \times L$ square of pixels, located anywhere on the $n \times n$ grid. Let $a_t(B)$ be the number of cream particles in B after t steps. Clearly

$$a_t(B) = \sum_{(x,y)\in B} a_t(x,y).$$

So it follows from Claim 1 that $\mathrm{E}[a_t(B)] \leq L^2$.

Fix some constant G, say 10. Then call B "bad" if $a_t(B)$ differs from $\mathrm{E}[a_t(B)]$ by more than L^2/G. Suppose that at some time step t, no B is bad. Also, suppose we form a coarse-grained image by coloring each B one of G shades of gray, depending on the value of

$$\left\lfloor \frac{a_t(B)\,G}{L^2} \right\rfloor$$

(or we color B white if $a_t(B) > L^2$). Then it's clear that the resulting image will be correctable, by adjusting each color by ±1, to one where all the B's within the same row are assigned the same color—and furthermore, that color is simply

$$\left\lfloor \frac{\mathrm{E}[a_t(B)]\,G}{L^2} \right\rfloor.$$

If this happens, though, then the Kolmogorov complexity of the coarse-grained image can be at most $\log_2(n) + \log_2(t) + O(1)$. For once we've specified n and t, we can simply *calculate* the expected color for each B, and no color ever deviates from its expectation.

So our task reduces to upper-bounding the probability that B is bad. By a Chernoff bound, since $a_t(B)$ is just a sum of independent, 0/1 random variables,

$$\Pr\left[|a_t(B) - \mathrm{E}[a_t(B)]| > \delta\,\mathrm{E}[a_t(B)]\right] < 2\exp\left(-\frac{\mathrm{E}[a_t(B)]\,\delta^2}{3}\right).$$

Plugging in $L^2/G = \delta\,\mathrm{E}[a_t(B)]$, we get

$$\Pr\left[|a_t(B) - \mathrm{E}[a_t(B)]| > \frac{L^2}{G}\right] < 2\exp\left(-\frac{L^4}{3G^2\,\mathrm{E}[a_t(B)]}\right).$$

Since $\mathrm{E}[a_t(B)] \leq L^2$ from above, this in turn is at most

$$2\exp\left(-\frac{L^2}{3G^2}\right).$$

Now, provided we choose a coarse-grain size

$$L \gg G\sqrt{3\ln(2n^2)} = \Theta\left(G\sqrt{\log n}\right),$$

the above will be much less than $1/n^2$. In that case, it follows by the union bound that, at each time step t, with high probability *none* of the $L \times L$ squares B are bad (since there at most n^2 such squares). This is what we wanted to show.