Ab initio Algorithmic Causal Deconvolution of Intertwined Programs and Networks by Generative Mechanism[*]

Hector Zenil[1,2,3,4], Narsis A. Kiani[1,2,3,4], Allan A. Zea[1,4,5], Jesper Tegnér[2,3,6]
[1] Algorithmic Dynamics Lab, Centre for Molecular Medicine, Karolinska Institute, Stockholm, Sweden
[2] Unit of Computational Medicine, Department of Medicine, Karolinska Institute, Stockholm, Sweden
[3] Science for Life Laboratory, SciLifeLab, Stockholm, Sweden
[4] Algorithmic Nature Group, LABORES for the Natural and Digital Sciences, Paris, France
[5] Escuela de Matemática, Facultad de Ciencias, UCV, Caracas, Venezuela
[6] Biological and Environmental Sciences and Engineering Division, Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Kingdom of Saudi Arabia
{hector.zenil, narsis.kiani, jesper.tegner}@ki.se

Abstract

Complex data is usually produced by interacting sources with different mechanisms. Here we introduce a parameter-free model-based approach, based upon the seminal concept of Algorithmic Probability, that decomposes an observation and signal into its most likely algorithmic generative sources. Our methods use a causal calculus to infer model representations. We demonstrate the method ability to distinguish interacting mechanisms and deconvolve them, regardless of whether the objects produce strings, space-time evolution diagrams, images or networks. We numerically test and evaluate our causal separation methods and find that it can disentangle examples of observations from discrete dynamical systems, and complex networks. We think that these causal separating techniques can contribute to tackle the challenge of causation for estimations of better rooted probability distributions thereby complementing more limited statistical-oriented techniques that otherwise would lack model inference capabilities.

Keywords: Model generation; signal decomposition; segmentation; algorithmic renormalization; algorithmic image segmentation; graph partitioning; algorithmic machine learning; model generation; feature selection.

[*]An online implementation is available at `http://www.complexitycalculator.com`. Code in the Wolfram Language and R is available at `https://github.com/allgebrist/Causal-Deconvolution-of-Networks/`

1 Introduction

To extract and learn representations leading to generative mechanisms from data, especially without making arbitrary decisions and biased assumptions, is a central challenge in most areas of scientific research particularly in connection to current major limitations of influential topics and methods of machine and deep learning as they have often lost sight of the model component.

Classical information theory has provided rigorous ways to capture our intuitive notions regarding uncertainty and information, and made an enormous impact in doing so. One of the fundamental measures here is mutual information, which captures the average information contained in one variable about another, and vice versa. If we have two source variables and a target, for example, we can measure the information held by one source about the target, the information held by the other source about the target, and the information held by those sources together about the target. Any other notion about the directed information relationship between these variables, which can be captured by classical information-theoretic measures (e.g., conditional mutual information terms) is linearly redundant with those three quantities.

However, intuitively, there is strong desire to measure further notions of how this directed information interaction may be decomposed, e.g., how much information the two source variables hold redundantly about the target, how much each source variable holds uniquely, and how much information can only be discerned by synergistically examining the two sources together. These notions go beyond the traditional information-theoretic view of a channel serving the purpose of reliable communication, considering now the situation of multiple communication streams converging on a single target. This is a common situation in biology, and in particular in neuroscience, where, say, the ability of a target to synergistically fuse multiple information sources in a non-trivial fashion is likely to have its own intrinsic value, independently of reliability of communication.

The absence of measures for such decompositions into redundant, unique and synergistic information is arguably the most fundamental missing piece in classical information theory. Triggered by the formulation of the Partial Information Decomposition framework by Williams and Beer in 2010, the past few years have witnessed a concentration of work by the community in proposing, contrasting, and investigating new measures to capture these notions of information decomposition.

Typically, models encode features of data in statistical form and in single variables and merged models, even in cases where several data sources are involved. Quantitative measures to disentangle complex signals and methods to tell apart causal mechanisms from noise and (ir)relevant interactions are of broad interest in areas such as statistical mechanics, reverse engineering, network inference, data reconstruction and scientific discovery in general, being germane to the challenge of causal analysis. For example, the development of techniques for learning disentangled representations using probabilistic machine and deep learning methods has recently gained significant momentum [11]. More broadly, model-based representations from data is one of the main challenges in machine learning, artificial intelligence and causal discovery and inference. Here we introduce a framework based upon the theory of algorithmic probability, which in our formulation is capable of identifying different sources that may explain and provide different models for each of the possible causes of convoluted or intertwined data. Casual inference has been one of the most challenging problems in science. The debate about causality has not prevented the development of successful and mature mathematical and algorithmic frameworks, first in the form of classical statistics and today in the form of computability and algorithmic information theories. Based on these latest mature mathematical frameworks that are acknowledged to fully characterize the concept of randomness as opposed to causal (deterministic), we introduced a suite of algorithms [20] to study the algorithmic information dynamics of evolving systems, and also methods to reduce the dimensions of data [21] based on the same principles. Algorithmic data dimension reduction and algorithmic deconvolution are two challenges which can be viewed as opposite sides of the same coin. On the one hand, data reduction is achieved by finding elements that are considered redundant, using as a criterion their contribution to the algorithmic content of the description of the data. Such elements provide the basis for compression and they can therefore safely be removed. On the other hand, the algorithmic deconvolution by generative mechanisms operates by identifying the elements that are likely to be part of the same causal path of some production rule or generating mechanism. The deconvolution is thus able to decompose causes and pinpoint the different sources generating the observed data.

2 Notation and Background

2.1 Cellular automata

We use cellular automata as causal dynamical systems to illustrate the algorithm before demonstrating its capabilities on more sophisticated objects of a convoluted nature and its applications to objects such as complex networks.

A cellular automaton is a computer program that applies in parallel a global rule composed of local rules on a tape of cells with symbols (e.g. binary). Thoroughly studied in [22], Elementary Cellular Automata (or ECA) are one-dimensional cellular automata that take into consideration in their local rules the cell next to the centre and the centre cell.

Definition 2.1. A *cellular automaton* (or CA) is a tuple $\langle S, (\mathbb{L}, +), T, f \rangle$ with a set S of states, a lattice $\mathbb{L}$ with a binary operation $+$, a neighbourhood template T, and a local rule f.

The *set of states* S is a finite set with elements s taken from a finite alphabet $\sum$ with at least two elements.

Definition 2.2. The *neighbourhood template* $T = \langle \eta_1, ..., \eta_m \rangle$ is a sequence of $\mathbb{L}$. In particular, the neighbourhood of cell i is given by adding the cell i to each element of the template T: $T = \langle i + \eta_1, ..., i + \eta_m \rangle$. Each cell i of the CA is in a particular state $c[i] \in S$. A *configuration* of the CA is a function $c : \mathbb{L} \to S$. The *set of all possible configurations* of the CA is defined as $S_{\mathbb{L}}$.

As a discrete dynamical system, the *evolution of the CA* occurs in discrete time steps $t = 0, 1, 2, ..., n$. The transition from a configuration c_t at time t to the configuration $c_{(t+1)}$ at time $t + 1$ is induced by applying the local rule f. The local rule is to be taken as a function $f : S^{|T|} \to S$ which maps the states of the neighbourhood cells of time step t in the neighbourhood template T to cell states of the configuration at time step $t + 1$:

$$c_{t+1}[i] = f\left(c_t[i + \eta_1], ..., c_t[i + \eta_m]\right) \tag{1}$$

The general transition from configuration to configuration is called the *global map* and is defined as: $F : S^{\mathbb{L}} \to S^{\mathbb{L}}$.

In the case of 1-dimensional CA it is common to introduce the *radius* of the neighbourhood template which can be written as $\langle -r, -r + 1, ..., r - 1, r \rangle$ and has length $2r + 1$ cells. With a given radius r the local rule is a function $f : \mathbb{Z}_{|S|}^{|S|^{(2r+1)}} \to \mathbb{Z}_{|S|}$ with $\mathbb{Z}_{|S|}^{|S|^{(2r+1)}}$ rules. *Elementary Cellular Automata (ECA)* have a radius $r = 1$ (closest neighbours), having the neighbourhood template $\langle -1, 0, 1 \rangle$, meaning that the neighbourhood comprises a central cell. From this it follows that the rule space for ECA contains $2^{2^3} = 256$ rules.

Enumeration of ECA rules: It is common to follow the lexicographic ordering scheme introduced by Wolfram [22]. According to this encoding, the 256 ECA rules can be encoded by 8-bits.

2.2 Causation and Algorithmic Probability

Algorithmic complexity [5, 2] is at the core of the challenge of complexity in discrete dynamical systems, as it involves finding the most statistically likely generating mechanism (computer program) that produces a set of given data. Formally, the algorithmic complexity (also known as Kolmogorov-Chaitin complexity) is the length of the shortest computer program that reproduces the data from its compressed form when running on a universal Turing machine.

We apply the so-called Coding Theorem Method (CTM) and Block Decomposition Method (BDM) as introduced in [3, 9, 13, 19] based on the seminal concept of Algorithmic Probability [10, 6] to estimate algorithmic complexity [5, 2].

3 Methods and deconvolution algorithm

3.1 Graph complexity

The concept of *Algorithmic Probability* (and associated Levin's semi-measure and Universal Distribution) has been introduced as a method for approximating algorithmic complexity based on the frequency of the patterns occurring in the adjacency matrix of a network. The measure applied to labelled graphs has been proven to be an tight upper bound of the algorithmic complexity of unlabelled graphs and therefore quite invariant to particular adjacency matrix choice [15].

More precisely, the algorithmic probability [10, 6, 2] of a subgraph $H \subseteq G$ is a measure of algorithmic probability based on the frequency of a random computer program p producing H when run on a 2-dimensional tape universal (prefix-free[1]) Turing machine U also referred to as a *Turmite*. That is, $m(G) = \sum_{p:U(p)=H\subseteq G} 1/2^{|p|}$.

The probability semi-measure $m(G)$ is related to algorithmic complexity $C(G)$ in that $m(G)$ is at least the maximum term in the summation of programs $m(G) \geq 2^{-C(G)}$, given that the shortest program carries the greatest weight in the sum. The Coding Theorem establishes the connection between $m(G)$ and $C(G)$ as ([6]): $|-\log_2 m(G) - C(G)| < c$ (Eq. 1), where c is some fixed constant, independent of s. The theorem implies that one can estimate the algorithmic complexity of a graph from the frequency of production from running random programs and applying the Coding theorem: $C(G') = -\log_2 m(G) + O(1)$. The Coding theorem establishes that graphs produced with lower frequency by random computer programs have higher algorithmic complexity, and vice versa.

The BDM of a graph thus consists in decomposing the adjacency matrix of a graph into subgraphs of sizes for which complexity values have been estimated, then reconstructing an approximation of the algorithmic complexity of the graph by adding the complexity of the individual pieces according to the rules of information theory, as follows:

$$C(G) = \sum_{(r_u,n_u)\in Adj(G)_{d\times d}} \log_2(n_u) + C(r_u) \tag{2}$$

[1]The group of valid programs forms a prefix-free set (no element is a prefix of any other, a property necessary to keep $0 < m(G) < 1$).

where $Adj(G)_{d\times d}$ represents the set with elements (r_u, n_u), obtained when decomposing the adjacency matrix of G into all subgraphs of size d contained in G. In each (r_u, n_u) pair, r_u is one such submatrix of the adjacency matrix and n_u its multiplicity (number of occurrences). As can be seen from the formula, repeated subgraphs only contribute to the complexity value with the subgraph BDM complexity value once plus a logarithmic term as a function of the number of occurrences. This is because the information content of subgraphs is only sub-additive, as one would expect from the growth of their description lengths. Applications of $m(G)$ and $C(G)$ have been explored in [3, 9, 8, 13], and include applications to graph theory and complex networks [12] and [13] where the technique was first introduced.

The only parameters used for the application of BDM, as suggested in [19], is to set the overlapping of the decomposition to the maximum 12 bits for strings and 4 square bits for arrays given the current best CTM approximations [9] from an empirical distribution based on all Turing machines with up to 5 states, and no string/array overlapping in the decomposition for maximum efficiency (as it runs in linear time) and for which the error (due to boundary conditions) has been shown to be bounded [19].

However, the algorithm introduced here is independent of the method used to approximate algorithmic complexity such as BDM. BDM assigns an index associated with the size of the most likely generating mechanism producing the data according to Algorithmic Probability [10]. BDM is capable of capturing features in data beyond statistical properties [19, 16] and thus represents an improvement over classical information theory. Because finding the program that reproduces a large object is computationally very expensive– even to approximate–BDM finds short candidate programs using another method [3, 9] that finds and reproduces fragments of the original object and then puts them together as a candidate algorithmic model of the whole object [19, 13]. These short computer programs are effectively model candidates explaining each fragment, with the long finite sequence of short models being itself a generating mechanistic model.

The aim of the deconvolution algorithm is to break a dataset into groups that do not share certain features (essentially causal clustering and algorithmic partition by probable generative mechanism, completely different from traditional clustering and partition in machine learning approaches). Usually these characteristics are a parameter to maximize, but ultimately the purpose is to distinguish components that are generated similarly from those that are generated differently. In information-theoretic terms the question is therefore as follows: What are the elements (e.g. nodes or edges) that can break a network into the components that maximize their algorithmic information content, that is, those elements that preserve the information about the underlying programs generating the data?

Let G be a graph and let $E = E(G)$ denote its set of edges. Let $G\backslash e$ denote the graph obtained after deleting an edge e from G. The *information contribution* of

e to G is given by $I(G, e) := C(G) - C(G\backslash e)$. A positive information contribution corresponds to information loss and a negative contribution to information gain. Here we wish to find the subset $F \subseteq E$ such that the removal of the edges in F disconnects G into N components and minimises the loss of information among all subsets of edges, i.e. the subset such that $I(G, F) \leq I(G, S)$ for all $S \subseteq E$. Let us denote the number of connected components of G by $k(G)$. Algorithm 1 allows us to obtain the subgraph $(V, E\backslash F)$ subject to the above conditions. The desired subset of edges is then given by $F = E(G)\backslash E(\textsc{Deconvolve}(G, N))$.

Algorithm 1 Causal deconvolution for networks

```
 1: function Deconvolve($G, N$), $1 \leq k(G) \leq N \leq |V(G)|$
 2:     while $k(G) < N$ do
 3:         $\mathit{infoloss} \leftarrow \varnothing$
 4:         // for each edge $e$
 5:         for $e \in E(G)$ do
 6:             if $I(G, e) > 0$ then
 7:                 // store the information loss after deleting $e$ into $\mathit{infoloss}$
 8:                 $\mathit{infoloss} \leftarrow \mathit{infoloss} \cup \{I(G, e)\}$
 9:         // calculate the minimal information loss across all edges
10:         $\mathit{minloss} \leftarrow \min(\mathit{infoloss})$
11:         // remove all edges with contribution $= \mathit{minloss}$ from $G$
12:         $G \leftarrow G\backslash\{e \in E(G) : I(G, e) = \mathit{minloss}\}$
        return $G$
```

The only parameter that Algorithm 1 requires is the number of components into which an object will be decomposed. However, there is a natural way to find the optimal terminating step and therefore the number of maximum possible components that minimize the sum of the lengths of the candidate generating mechanisms thereby truly making the algorithm parameter-free as it is not required to have a preset number of desired components.

Before introducing the terminating criterion (c.f. next section) for the number of components, let's analyse what it might mean for two components s_1 and s_2 to have same algorithmic information content $C(s_1) = C(s_2)$. Clearly that subcomponents s_1 and s_2 have the same algorithmic complexity (an integer—or a real value if using AP-based BDM—indicating the size of the approximated minimal program) does not imply that the two components are generated by exactly the same generating mechanism. However, because of the exponential decay of the algorithmic probability of an increasingly random object, we have it that the less random it is, the exponentially more likely it is that the underlying mechanism will be the same (see Fig. 5C). This is because there are exponentially fewer short programs than long ones. For example, in the extreme case of connected graphs, we have it that the complete graph denoted by K_n has the smallest possible algorithmic complexity $\sim \log(n)$. If $C(s_1) = C(s_2) \sim \log(n)$ then s_1 and s_2 are, with extremely high probability, generated by the same algorithm that generates either the complete graph or the empty graph (with same lowest algorithmic complexity as it requires no description other than either all nodes connected or all nodes disconnected). Conversely, if $C(s_1) = C(s_2)$ but $C(s_2)$ and $C(s_1)$ depart from $\log(n)$ (and approximate algorithmic randomness) then the likelihood of being generated by the same algorithm exponentially vanishes. So the information regarding both the algorithmic complexity of the components and their relative size sheds light on the candidate generating mechanisms and is less likely to coincide 'by chance' for non-trivial cases.

3.1.1 Algorithm terminating criterion

The immediate question is where we should stop breaking down a system into its causal components. The previous section suggests a terminating criterion. Let S be the object which has been produced by N mostly independent generative mechanisms. We decompose S into n parts $s_1, \ldots, s_n$ in such a way that each s_i, $i \in \{1 \ldots n\}$ has an underlying generating mechanism found by running the algorithm iteratively for increasing n, but after each iteration we calculate the minimum of the differences in algorithmic complexity among all subcomponents. The algorithm should then stop where the number of subcomponents is exactly N when the sum of the lengths—the estimated algorithmic complexity—of each of the programs will diverge from the expected $log(N)$ because the length of the individual causal mechanisms producing each new component will be breaking a component that could previously be explained by the causal mechanism at a previous iteration of the algorithm.

As a trivial example, let's take the string 1^n, where S^n means that the pattern S is repeated n times. After application of the algorithm, the terminating criterion will suggest that 1^n cannot be broken down into smaller segments, each with a different causal generating mechanism, whose total length sums will be shorter than the length of the generating mechanism producing 1^n itself. This is because the sum of the length of the shortest programs $\sum_i |p_i|$ running on a universal Turing machine generating segments of 1^n of length $m_i < n$ each, such that the concatenation $\cup_{i=1} p_i = 1^n$, will be strictly greater than $C(1^n)$, given that each p_i halting criterion will require $i \log m_i$ bits more than $C(1^n)$.

In the case of Fig. 2, the terminating criterion retrieves $N = 3$ components from the two interacting ECA (rule 60 and 110). This does not contradict the fact that we started from two generating mechanisms, because there are three clear regimes that are actually likely to be reproducible by three different generating mechanisms, as suggested by the deconvolution algorithm itself, and as found in [7], where it has been shown that rule 110 can be emulated by the composition of two simpler ECA rules (rules 51 and 118). As seen in Fig. 2, among the possible causal partitions, $N = 2$ successfully deconvolves ECA rule 60 from rule 110 on the first run, with a stronger difference than the difference found between $N = 3$ components when breaking rule 110 into its two different regimes.

3.1.2 Time complexity

The algorithmic for network deconvolution that we have introduced in this section runs in polynomial-time in the general case for a small number of components to deconvolve. Let M denote the number of edges of the graph G. The brute force algorithm for this problem searches the edge such that its removal minimises the loss of information and deletes it, repeating this process for all edges of G until N subcomponents are reached, which has a worst-case time complexity of $O(M^2)$. Algorithm 1 is different from the brute force approach in that edges with equal minimal contribution to the loss of information are not deleted sequentially but all at once. While Algorithm 1 also has a worst-case time complexity of $O(M^2)$, this slight modification makes it more optimal in some cases.

4 Numerical experiments

4.1 Decomposition of sequences and space-time diagrams

In this section, we will test the suggested algorithm on different types of objects, in order to show its applicability and power. We start with the simplest version of an object which conveys information, a string, and move later to consider richer objects such as networks.

We will use different programs to produce different parts of a string, that is a program p to generate segment s_1 and program p' to generate segment s_2 put next to each other. Clearly the string has been generated by two generating mechanisms (p and p'). Now we use the algorithm to deconvolve the string and find the number of generating mechanisms and most likely model mechanisms (the program themselves) inducing a form of *algorithmic partition* based on the likelihood of each segment to be produced by different generating mechanisms.

Figs. 1A-E illustrate how strings that have short generating mechanisms are significantly and consistently more sensitive to perturbations. The resulting string is 010101010101010101010101010101010101010101010101010111010010101010000000 10011001111001100000011100110 with the colours corresponding to the parts suggested by the different regimes, according to their algorithmic contribution and the segment's resilience to perturbations (by deletion and replacement) to the original string. Behind every real number approximating the algorithmic complexity of a

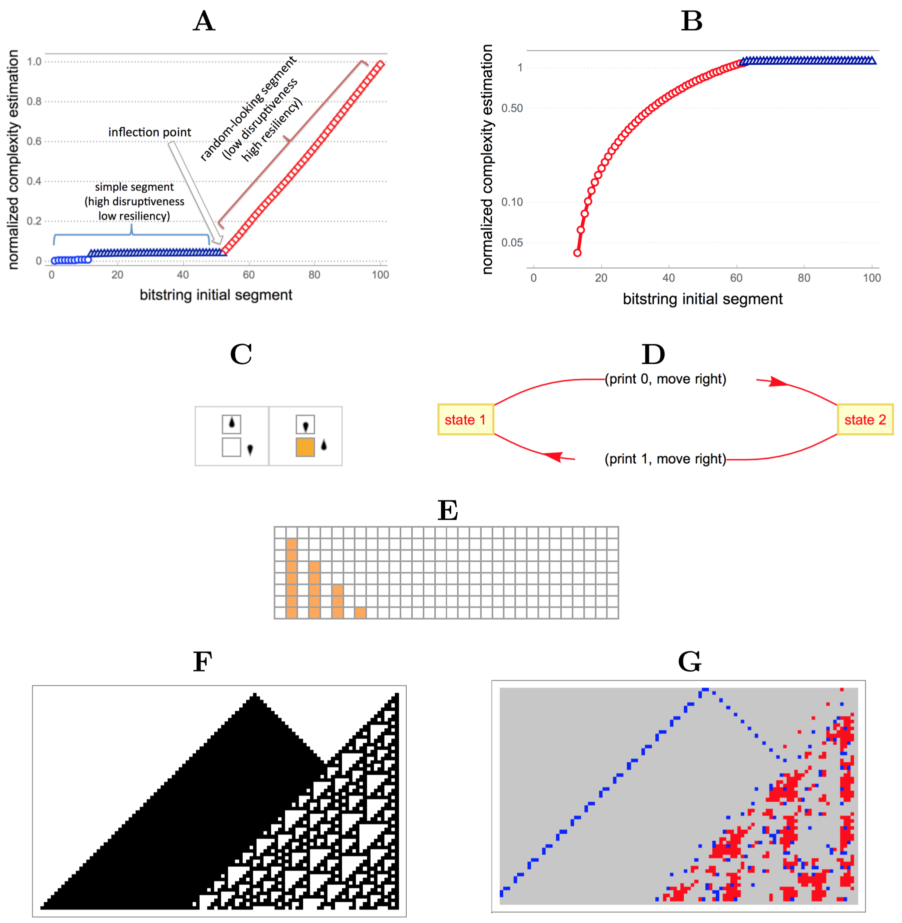

Figure 1: Proof of concept applied to a binary string composed of two segments with different underlying generating mechanisms (computer programs). A: Log plot of complexity estimation of a regular segment (blue) consisting of the repetition of '01' 25 times followed by a random-looking segment (red). B: Log plot reversing the order of A yet preserving the qualitative behaviour of the different segments. C: The code of the smallest generating program (a non-terminating Turing machine) depicted visually (states are arrows in different directions) producing the string of 01^n for any n (0 is white and 1 is orange) starting from a blank tape as shown in the space-time diagram (E). D: the same computer program as a state diagram. F: Interacting programs with different generating mechanisms (ECA rules 255 v 110) running for 60 steps. G: Algorithmic information footprint, every pixel is deleted and its original contribution to the whole quantified and coloured accordingly. If grey, then it makes the lowest contribution, blue represents a low contribution and red the highest contribution (randomness).

string there is the discovery of a large set of generating programs when using the Algorithmic Probability (AP)-based measure BDM producing the object.

We not only could find the number of mechanisms correctly (Figs. 1A and B) but the candidate programs (which for this trivial example are exactly the original) that generate each segment (Figs. 1C-E) by way of seeking for the shortest computer programs in a bottom up approach (see [3, 9, 13, 19]). Finding the shortest programs is, however, secondary, because we only care about the different explanatory power that different programs have to explain the data in full or in part pinpointing the different causal nature of the segments and helping in the deconvolution of the original observation.

Figs. 1C-E depict the computer program (a non-terminating Turing machine) that is found when calculating the BDM of the 01^n string. The BDM approximation to the algorithmic complexity of any 01^n string is thus the number of small computer programs that are found capable of generating the same string or, conversely (via the algorithmic Coding theorem, see [3, 9, 13]), the length of the shortest program producing the string. For example, the string 01^n was trivially found to be generated by a large number of small computer programs (in Fig. 1C,D depicted a non-terminating Turing machine with E its output) using our algorithmic methods (as opposed to, e.g., using lossless compression, which would only obfuscate the possible generating model) with only two rules out of 2×2 rules for the size of Turing machine with only two states and two symbols and no more, thus of very low algorithmic complexity compared to, e.g., generating a random-looking string that would require a more complex (longer) computer program. The computer program of a truly random string will grow in proportion to the length of the random string, but for a low complexity string such as 01^n, repeated any number of times n, the length of the computer program is of (almost) fixed size, growing only by $log(n)$ if the computer program is required to stop after n iterations. In this case 01^n is a trivial example with a strong statistical regularity whose low complexity could be captured by applying Shannon entropy alone on blocks of size 2.

Figs. 1 F and G illustrate how the algorithm can separate regions produced by generating mechanisms of different algorithmic information content by observing their space-time dynamics, thereby contributing to the deconvolutin of regions that are produced by different generating mechanisms. In this example both programs are sufficiently robust to not break down (see Sup. Inf.) when they interact with each other, with rule 110 prevailing over 255. Yet, in the general case it is not always easy to tell these mechanisms apart. In more sophisticated examples such as in Figs. 2 D to E, we see how the algorithm can break down contiguous regions separating an object into two major components corresponding to the different generating computer programs that are intertwined and actively interacting with each other. The experiment was repeated 20 times with programs with differing qualitative (e.g. Wolfram class) behaviour.

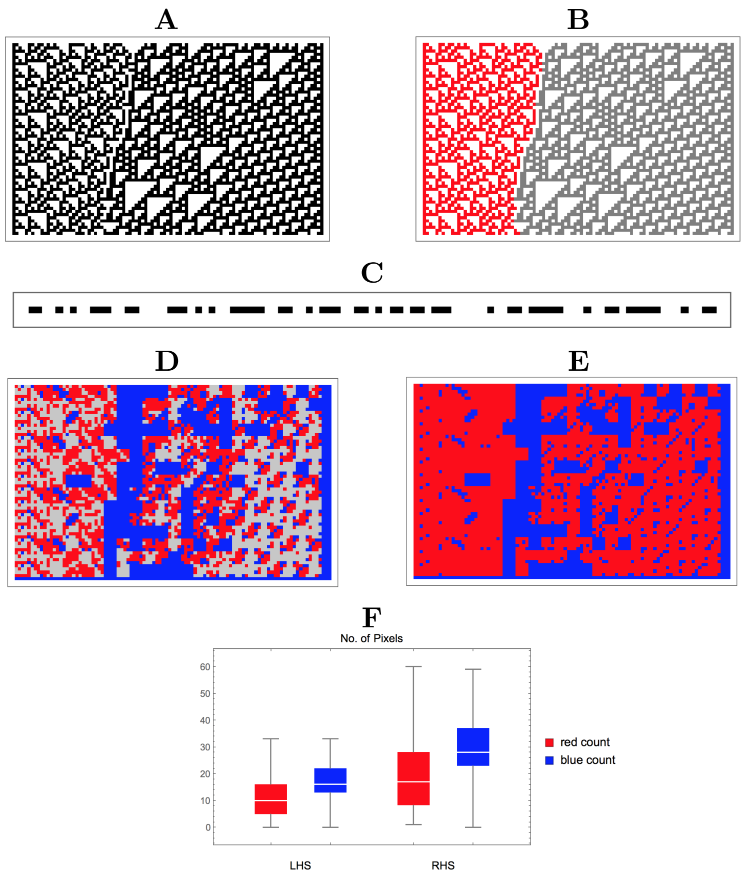

Figure 2: A: The output of two different intertwined programs (ECA rules 60 and 110) with qualitatively complex output behaviour (11 to 60 steps depicted here, from a random initial condition) interacting with each other (one of which has been proven to be Turing-universal and the other has also been conjectured to be universal [22]), each producing structures of a similar type that, from an observer's perspective, are difficult to distinguish (see Subfigure C) as is artificially done in B (knowing which pixel is generated by which rule). C: What an observer of the last runtime would see in the form of a stream of bits with no clear statistical distinction. D: The algorithm pinpoints the regions of neutral, positive and negative, with the contiguous largest blue component segmenting the image into two components. E: only negative vs positive causal contributions where both Shannon entropy and a popular lossless compression algorithms fail (see SI). F: Sanity check/validation: Statistically significant quantitative differences among the parts after application of the algorithm as illustrated in E among apparently weak qualitative differences as illustrated in Subfig. A.

Fig. 2 F demonstrates that perturbations to regions in red are considered to have a more random effect and are thus by themselves less algorithmically complex (random) versus simple. When regions are of the same algorithmic complexity they are likely to be generated by similar algorithms, or more precisely, algorithms that are of similar minimal length. Regions in blue move the space-time evolution away from randomness and are themselves more algorithmically random. Blue structures on the left hand side correspond to large triangles occurring in ECA rule 110 that are usually used to compute and transfer information in the form of particles. However, triangular patterns transfer information in a limited way because their light cone of influence reduces at the greatest possible speed of the automaton, and they are assigned an absolute neutral information value. Absolute neutral values are those closest to 0. Once separated, the two regions have clearly different algorithmic characteristics given by their causal perturbation sensitivity, with the right hand side being more sensitive to both random and non-random perturbations. Moreover, Fig. 2 F shows results compatible with the theoretical expectation and findings in [20] where a measure of reprogrammability associated with the number and magnitude of elements that can move a dynamical system towards or away from randomness was introduced and shown to be related to fundamental properties of the attractor space of the system.

4.2 Graph and network deconvolution

Classification can usually be viewed as solving a problem which has an underlying tree structure according to some measure of interest. One way to think of optimal classification is to discover a tree structure at some level of depth, with tree leaves closer to each other when such objects have a common or similar causal mechanism and for which no feature of interest has been selected. Fig. 3 illustrates how the algorithm may data, in this case starting from a trivial example that breaks complete K-ary trees. Traditionally, partitioning is induced by an arbitrary distance measure of interest that determines the connections in a tree, with elements closer to a cluster centre connected by edges. The algorithm breaks the trees (see Fig. 3) into as many components as desired by iterating the algorithm until the number of desired components is obtained or the terminating criterion is applied (c.f. Subsection 3.1.1). Figs. 3A,B provide examples illustrating how to maximize topological symmetry. The algorithm can be applied, without loss of generalization, to any non-trivial graph, as in Figs. 3C,D or on any dataset for that matter.

Figs. 4 illustrate the algorithm and terminating criterion starting from an artificial graph composed by several graphs (2 simple and one E-R random: a small E-R random graph connected to a star graph and to a complete graph). The graph can be successfully decomposed by algorithmic probability (see Figs. 4B and D) by identifying the likelihood of an edge to be produced by the same mechanism by

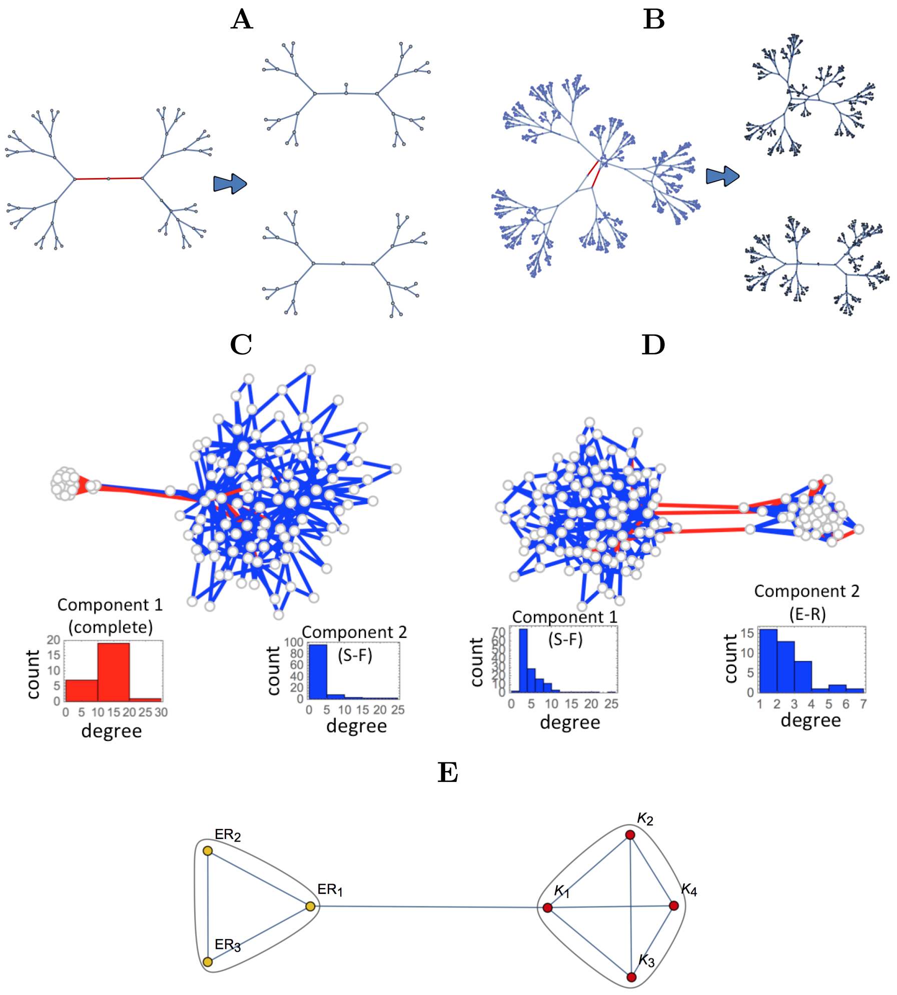

Figure 3: A,B: Forced deconvolution of a tree by minimization of graph algorithmic information loss thereby maximizing causal resemblance of the resultant components (hence causal clustering). Depicted are the components of a K-ary trees of size 6 (A) and 10 (B) and their resulting graphs after one iteration of the deconvolution algorithm. C: Deconvoluting a graph composed by a complete graph and a scale-free (S-F) network generated by preferential attachment and randomly connected between them. Negative edges break down the graph into components corresponding to the different underlying generating mechanisms. D: Deconvolution of a random graph (E-R) and a scale-free (S-F) network. E: The algorithm first separates the subcomponents with the largest algorithmic difference, followed by other subcomponents.

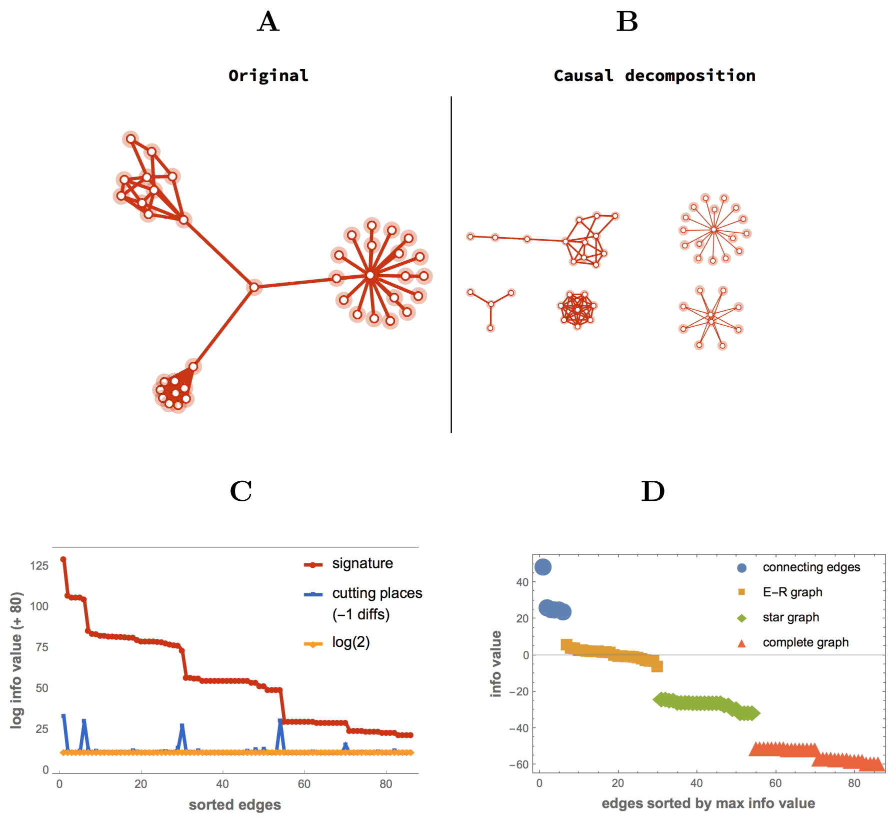

Figure 4: A: Synthetic graph composed by 3 subgraphs with different topology. B: The causal decomposition in smaller components separating the graph into smaller spanning subgraphs, further partition starts breaking subgraphs whose resultant components' algorithmic complexity are too close to each other indicating the terminating criterion. C: The *information signature* (red circle) illustrates the distribution of information values for each edge (x-axis) from the original graph (A), a line of differences of consecutive values (blue square) from the signature indicating the largest values producing a natural separation (the peaks signal the edges to delete) with, in this case, four values clearly standing out beyond the log(2) line (yellow rhombus) breaking the signature corresponding to each subgraph forming with high accuracy thereby deconvolving the original graph (A) into the spanning graphs that are most likely generated by the same causal/algorithmic source.

virtue of being close to each other in the information contribution (which theoretically should be removed by only log(2) if it follows the normal evolution of the same process), hence what we call causal separation/partition and clustering. Fig. 4D shows the distribution of all edges coloured by its graph membership and almost perfectly corresponding to the different pieces separated by the largest differences as shown in Fig. 4A.

The same task using classical information theory (Shannon entropy) is shown not to be sensitive enough (see Sup. Inf.), and a popular lossless compression algorithm (Compress based on LZW) provided a noisy approximation (see Sup. Inf.) to the results obtained by using the Block Decomposition Method, as defined in [19], whose description is provided in the Sup. Inf.

Figs. 3C-E illustrate how randomly connected graphs with different topologies can be broken into their respective generative mechanisms. Fig. 3C is a complete graph of size 20 randomly connected by 3 edges to a scale-free graph of size 100. The graphs are generated by different mechanisms, one is a small program that, given a number N of nodes, produces a graph with all nodes connected to all other $N-1$ nodes and has a program of small length that grows only by $\log N$ [15]. The scale-free network is generated by the canonical preferential attachment algorithm with two edges per node and requires a slightly longer algorithm that grows by $\log N + c$ [15] where c is a small constant accounting for the pseudo-random choice of attachment nodes. The algorithm breaks the graphs into two components, each of which corresponds to the graphs with different degree distribution (depicted below each case) associated with its generating mechanism. This is because $|P(G_1)| + |P(G_2)| + \ldots + |P(G_n)| + |P(e_{G_i})| > |P(G_1 G_2 \ldots G_n)|$ for any G_i, where e_{G_i} is the set of edges randomly connecting G_i to G_j for any i and j for all G of low algorithmic complexity.

4.3 Robustness and limitations

Fig. 3 D illustrates a similar case to Fig. 3 C, but instead of a complete graph an Erdős-Rényi (E-R) graph with edge density 0.5 is produced and connected by 3 random edges to a scale-free network produced in the same fashion as in Fig. 3 C. Again, the algorithm was able to break it down into the two corresponding subgraphs. Fig. 3 D represents a test case to evaluate the effect of additive noise by connecting an E-R graph of increasing size and with an increasingly greater number of random edges.

Next we ask how much structure, if any, can be recovered/extracted when adding a random (E-R) graph to different types of structured networks. To this end, we conducted a series of numerical experiments shedding light on the limitations of the algorithm introduced here in the face of additive noise. The same results were obtained for the simpler case of connecting any complete graph of increasing size to any other graph such as E-R or S-F.

Fig. 5 shows the results from the experiments separating graphs, in this case a scale-free graph (S-F) from an Erdős-Rényi graph (E-R), the former generated by a Barábasi-Albert preferential attachment algorithm [1] and the latter produced

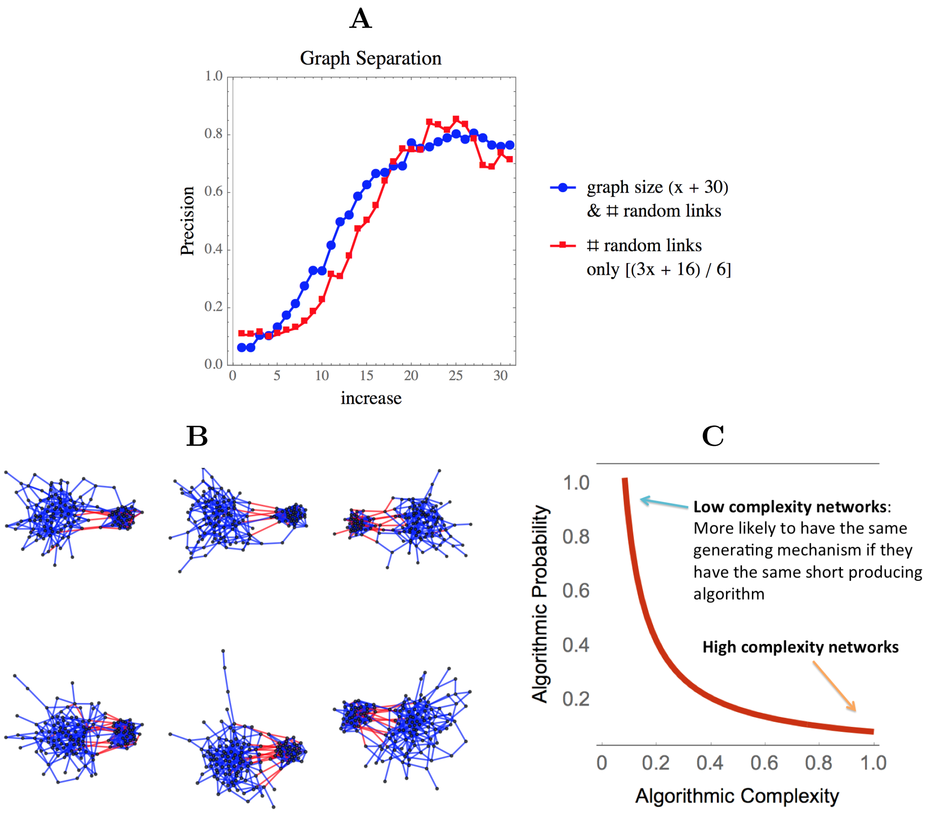

Figure 5: A: Causal deconvolution of graphs, a S-F graph (2 new links per added node) from an E-R (0.5 edge density) graph emulating noise (each point represents the average of 10 replicates). When the number of links increases as a function of the subgraph sizes, the separability is robust (red square markers) compared to increasing the number of random links only for graphs of fixed size (40 nodes), when successful separation is compromised, noise (E-R) and structure (S-F) being indistinguishable. B: A small sample of 6 graphs among the $10 \times 20 = 200$ graphs randomly connecting a S-F to an E-R graph. Success at identifying randomly connecting links is shown and was found to be very robust. C: Objects have exponentially greater chances of being produced by the same generating mechanism if they are of low algorithmic randomness and thus of high algorithmic probability.

by a pseudo-random generator. Fig. 5A quantifies the error and optimal signal-to-noise ratio for optimal deconvolution, testing the algorithm under additive noise both for fixed and growing size subcomponents. Fig. 5B shows links coloured in red as identified by the algorithm having the highest algorithmic information value when their removal sends the original composed system towards lower algorithmic information content, thereby telling apart the two subcomponents. Negative links are mostly on the side of the E-R graph. In other words, the S-F can be extracted with the greatest precision and a lower rate of false positives from the mix, which is to be expected given the random nature of the added links that connect the graphs, making them more like the E-R links than the S-F. If only the number of random links among graphs increases for fixed size graphs (Fig. 5B blue circle marks) a maximum precision of about 0.9 is reached before degradation. That is, at around 32.5% of the links randomly connecting the components. In other words, the algorithm is robust, telling apart noise from structure even after up to 0.325 (from the random links connecting the components) + 0.5 (from the E-R component) = 0.825, i.e. 82.5% of all links are random. On the other hand, the number of false positives is constant at about 5%, in the case shown in Fig. 5B all of the false positives (red links not connecting the two graphs with different topology) are inside the E-R graph and mostly nonexistent on the side of the less random S-F graph.

5 Remarks and conclusions

For some more sophisticated yet successful examples on which these new algorithmic methods may outperform applications of entropy in general (without access to true probability distributions) and other computable measures see [19], and [16] for a non-trivial example in which entropic measures fail (by offering divergent descriptions of the same evolving system). The generation of these models is key in our approach because the integer (or real) value assigned to a system as an estimation of its algorithmic complexity is nothing but a guiding index of the number of specific models found by our method that are capable of explaining and generating the data.

One possible objection is that any number of interacting rules can be thought of as a single rule producing an intertwined output because we known from Turing universality that any number of interacting programs can also be rewritten as a single program in a larger rule space (defined by state $\times$ symbol) incorporating all the behaviours together. That is, any computer program can be decomposed into one or more computer programs producing the same output. This means that separating programs and signals is, in some way, not fundamental. Fig. 2C-F shows, for example, that by iterating the deconvolution algorithm not only do the two main components of the image correspond to the two generating ECA rules, but a second application of the algorithm would produce a third or more components corresponding to further resilient features generated by the rules, which can be considered rules themselves within a smaller rule (state/symbol) space. However, in the deconvolved observations the interacting rule determining how two or more rules may interact effectively constitutes a third global rule to which the algorithm has no direct access or an apparent region in the observed window.

We have introduced and tested a parameter-free algorithm for causal deconvolution of interlaced and interacting mechanisms using fundamental concepts drawn from the theory of Algorithmic Probability and Algorithmic Information Theory. Our approach enables a parameter-free analysis of the deconvolution problem, since we have removed the need for pre-defined user-centric definitions of global properties or local rules (e.g. distance metrics) to determine the algorithm. Instead, relating our algorithm to the algorithmic information theory provides us with a fundamental metric. While the algorithm uses state-of-the-art algorithms to approximate algorithmic complexity (CTM and BDM), the algorithm and methods introduced here are independent of the approximating method chosen (e.g. lossless compression). However, the precision and accuracy is not. Yet, the algorithm is sufficiently robust to disentangle sophisticated intertwined causal mechanisms. This opens the possibility of parlaying these algorithmic methods into a more elaborate machine learning framework. For example, the extracted causal information can be used as a prior distribution for machine and deep learning techniques. This is useful because efficient techniques for pattern recognition are as a rule weak on inferring models, and thus ill-equipped to capture the underlying generative mechanisms and thereby produce predictive and prescriptive causal models that exceed the descriptive nature of current classical and modern statistical approaches.

Acknowledgements

H.Z. was supported by the Swedish Research Council (Vetenskapsrådet) grant No. 2015-05299.

References

[1] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, *Reviews of Modern Physics*, 74 (1): 47–97, 2002.

[2] G.J. Chaitin. On the length of programs for computing finite binary sequences *Journal of the ACM*, 13(4):547–569, 1966.

[3] J.-P. Delahaye and H. Zenil, Numerical Evaluation of the Complexity of Short Strings: A Glance Into the Innermost Structure of Algorithmic Randomness, *Applied Mathematics and Computation* 219, 63–77, 2012.

[4] E. Hermo-Reyes and J.J. Joosten, "Competing Cellular Automata" `http://demonstrations.wolfram.com/CompetingCellularAutomata/`, Wolfram Demonstrations Project, June 17, 2014.

[5] A.N. Kolmogorov. Three approaches to the quantitative definition of information, *Problems of Information and Transmission*, 1(1):1–7, 1965.

[6] L.A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory, *Problems of Information Transmission*, 10(3):206–210, 1974.

[7] J. Riedel and H. Zenil, Rule Primality and Compositional Emergence of Turing-universality from Elementary Cellular Automata, forthcoming, 2017. XXX

[8] F. Soler-Toscano, H. Zenil, J.-P. Delahaye and N. Gauvrit, *Correspondence and Independence of Numerical Evaluations of Algorithmic Information Measures*, Computability, vol. 2, no. 2, pp. 125–140, 2013.

[9] F. Soler-Toscano, H. Zenil, J.-P. Delahaye and N. Gauvrit, *Calculating Kolmogorov Complexity from the Frequency Output Distributions of Small Turing Machines*, PLoS One 9(5), e96223, 2014.

[10] R.J. Solomonoff, A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[11] N. Siddharth, B. Paige, J-W van de Meent, A. Desmaison, N.D. Goodman, P. Kohli, F. Wood, P.H.S. Torr, Learning Disentangled Representations with Semi-Supervised Deep Generative Models, arXiv:1706.00400 [stat.ML], 2017.

[12] H. Zenil, F. Soler-Toscano, K. Dingle and A. Louis, Graph Automorphisms and Topological Characterization of Complex Networks by Algorithmic Information Content, Physica A: Statistical Mechanics and its Applications, vol. 404, pp. 341–358, 2014.

[13] H. Zenil, F. Soler-Toscano, J.-P. Delahaye and N. Gauvrit, *Two-Dimensional Kolmogorov Complexity and Validation of the Coding Theorem Method by Compressibility*, 2013.

[14] H. Zenil, N.A. Kiani and J. Tegnér, Quantifying Loss of Information in Network-based Dimensionality Reduction Techniques, Journal of Complex Networks 4, 342–362, 2016.

[15] H. Zenil, N.A. Kiani and J. Tegnér, Methods of Information Theory and Algorithmic Complexity for Network Biology, *Seminars in Cell and Developmental Biology*, vol. 51, pp. 32-43, 2016.

[16] H. Zenil, N.A. Kiani and J. Tegnér, Low-Algorithmic-Complexity Entropy-deceiving Graphs, *Physics Reviews E.* 96, 012308, 2017.

[17] H. Zenil, Compression-based Investigation of the Dynamical Properties of Cellular Automata and Other Systems, *Complex Systems*, 19(1), pages 1–28, 2010.

[18] H. Zenil and E. Villarreal-Zapata, Asymptotic Behaviour and Ratios of Complexity in Cellular Automata Rule Spaces, *International Journal of Bifurcation and Chaos*, vol. 13, no. 9, 2013.

[19] H. Zenil, S. Hernández-Orozco, N.A. Kiani, F. Soler-Toscano, A. Rueda-Toicen, A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity, arXiv:1609.00110 [cs.IT], 2016.

[20] H. Zenil, N.A. Kiani, F. Marabita, Y. Deng, S. Elias, A. Schmidt, G. Ball, J. Tegnér, An Algorithmic Information Calculus for Causal Discovery and Reprogramming Systems, 2017. BioArXiv DOI: https://doi.org/10.1101/185637

[21] H. Zenil, N.A. Kiani, J. Tegnér, Data Reduction and Network Sparsification by Minimal Algorithmic Information Loss, arXiv XXX

[22] S. Wolfram, *A New Kind of Science*, Wolfram Media, Champaign IL., 2002.

Supplementary Information

5.1 Interacting programs

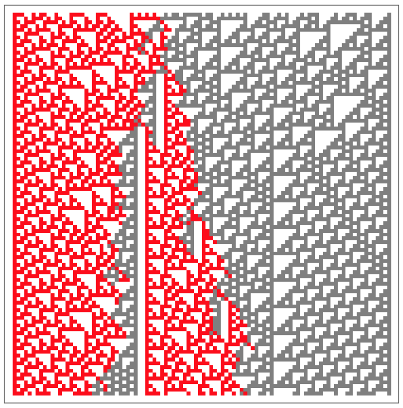

Figure 6: Interacting programs such as cellular automata require us to define how the interaction happens by, e.g., deciding what set of rules apply at the intersection. For instance, one of the two sets of local rules or a 3rd set of rules effectively defining a super cellular automaton that most likely is another cellular automaton in a larger rule-space (requiring more states to define the two sub-cellular automata and the interaction). Depicted here is the type of interaction we explored, and this case shows the richness of such possible interactions where both rules 'spill over' each other.

The qualitative behaviour of each program can heuristically be identified by what is known as its Wolfram class, which in turn has been formalized using tools and methods from algorithmic complexity in [17, 18]. Informally, Wolfram class 1 represents evolutions of programs that converge to a simple fixed configuration, exemplars of Wolfram class two converge to repetitive simple behaviour, those of Wolfram class 3 produce unbounded apparently statistically random behaviour and exemplars of Wolfram class 4 reproduce apparently open-ended persistent structures. None of what has been introduced here depends on this behavioural characterization based on different heuristics, and it is thus in no way fundamental to the results reported.

The interaction rule is interaction rule number 1 according to the enumeration open-source program on the Wolfram Demonstrations website publicly available at `http://demonstrations.wolfram.com/CompetingCellularAutomata/` [4] determining the way in which the local rules from each global ECA rule will be dictated and applied at their intersection.

5.2 Graph generation

The graphs used throughout this paper were generated using the Wolfram Language on the Mathematica platform using the function RandomGraph[] with uniform distribution (UniformGraphDistribution[]) for Erdős-Rényi graphs and a scale-free distribution (BarabasiAlbertGraphDistribution[]) for the scale-free networks constructed by starting from a cycle graph of size 3 and a vertex of k edges added at each step according to the preferential attachment algorithm [1] following a distribution proportional to the vertex degree. All experiments were replicated and averaged from a set of at least 20 instances.

5.3 Comparison with entropy and lossless compression

Fig. 7 shows the results obtained by using classical information theory (Shannon entropy) and one of the most popular lossless compression algorithms (Compress) based on LZW as an approximation to algorithmic (Kolmogorov-Chaitin) complexity instead of BDM. Because all values collapse into a single value for entropy, the colours displayed are the result of an artificial sorting of the pixels based on their indices, from top to bottom. Compression is a lower-quality approximation of what we reported in Figs. 1C-F, where the reported algorithm based on the BDM is clearly an improvement.

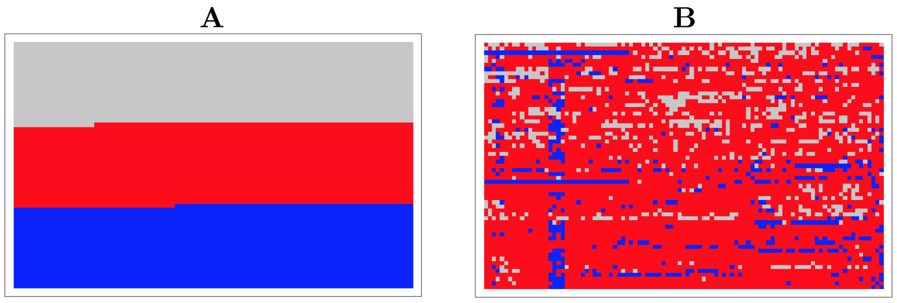

Figure 7: Shannon entropy (A) and lossless compression (Compress) underperform, not being sensitive enough in performing the same task reported in Figs. 1C-F.