Algorithmic causal structure emerging through compression

Liang Wendong WENDONG.LIANG@TUE.MPG.DE
Max Planck Institute for Intelligent Systems, Tübingen, Germany
Simon Buchholz SIMON.BUCHHOLZ@TUE.MPG.DE
Max Planck Institute for Intelligent Systems, Tübingen, Germany
Bernhard Schölkopf BS@TUE.MPG.DE
Max Planck Institute for Intelligent Systems & ELLIS Institute, Tübingen, Germany

Editors: Biwei Huang and Mathias Drton

Abstract

We explore the relationship between causality, symmetry, and compression. We build on and generalize the known connection between learning and compression to a setting where causal models are not identifiable. We propose a framework where causality emerges as a consequence of compressing data across multiple environments. We define algorithmic causality as an alternative definition of causality when traditional assumptions for causal identifiability do not hold. We demonstrate how algorithmic causal and symmetric structures can emerge from minimizing upper bounds on Kolmogorov complexity, without knowledge of intervention targets. We hypothesize that these insights may also provide a novel perspective on the emergence of causality in machine learning models, such as large language models, where causal relationships may not be explicitly identifiable.

Keywords: algorithmic causality, compression, symmetry, Kolmogorov complexity

1. Introduction

The case has been made that learning and compression are intimately related (Blumer et al., 1987; MacKay, 2003; Grünwald, 2007; Vapnik, 1999, Sec. 4.6): both are made possible by regularities in datasets. In the present paper, we seek to extend this connection beyond predictive machine learning. Such settings are often studied using causal models such as structural causal models (SCMs) and causal Bayesian networks (CBNs) (Pearl, 2009). We are particularly interested in settings where data is non i.i.d. and originates from multiple environments which share (most of the) mechanisms or causal conditionals. This assumption is termed sparse mechanism shift (SMS) (Schölkopf et al., 2021). Intuitively, SMS helps us when jointly compressing models learned across multiple environments, since shared mechanisms then need only be compressed once. However, it is not trivial to integrate this into the known compression framework. This serves as a motivation for our paper, with the goal of working towards a rigorous framework of compression in which causality emerges as a regularity bias.[1]

As a starting point, we observe that most of the previous literature on identifiability in causal discovery and causal representation learning can be rephrased in a classical compression framework: There is (up to tolerable ambiguities) a unique model under which the data has maximal likelihood, equivalently, a unique model whose distribution has minimal cross-entropy with the data distribution. A common criticism of causality research is that identifiability generally requires

[1] An extended version of our paper is available at: https://arxiv.org/abs/2502.04210

strong assumptions. We are interested in cases when those assumptions do not hold, and investigate what we can still say about causality. As a motivation we remark that if the only well-defined notion of causality were to build upon identifiability subject to unrealistic assumptions, why could humans and animals possess reliable causal knowledge of aspects of the world, and how should we formalize this knowledge despite the issues of non-identifiability?

Contributions

- We discuss the relationship between identifiability (in causal discovery and causal representation learning) and compression (§ 2).
- To the best of our knowledge, our work is the first rigorous treatment of principled decisions (rather than only non-identifiability) of causal arrows with no constraints on distribution classes and no knowledge of intervention types or targets in a non-Bayesian perspective, with a weaker definition of causality (§ 3, Definition 3, Definition 38).
- Under the settings where minimum cross-entropy cannot identify causal arrows, we use a more general notion of compression, i.e., Kolmogorov complexity, to carry out model selection over the algorithmic causal models. (§ 4) We provide computable upper bounds on Kolmogorov complexity (i.e., finite codebook complexity (Definition 14)) under certain non-universal Turing machines, i.e., universal finite codebook computers (UFCC, Definition 13).
- We prove that under some UFCCs, models using causal factorizations and models with the sparsest mechanism shifts are preferred by minimizing the finite codebook complexity. (§ 5, Proposition 46) This means that algorithmic causality emerges as a by-product when minimizing an upper bound of Kolmogorov complexity. We further show that some UFCCs align simplicity (i.e., short coding length) with symmetries such as invariance or equivariance under group actions (Proposition 18).

2. Identifiability in causality and its relation with compression

In this section, we review the general relation of compression and identifiability in causality, and the limitations of both two notions. We focus on CBNs since Pearl (2009) shows that identifying CBNs is strictly easier than identifying SCMs, and hence all difficulties regarding identifiability in CBNs also exist in SCMs. An **observational CBN model** is a tuple $(G, \mathbb{P})$ where $G = (V, E)$ is a directed acyclic graph (DAG), and the distribution $\mathbb{P}$ is Markov relative to G, i.e., $\mathbb{P}(X) = \prod_i \mathbb{P}(X_i | X_{\mathrm{pa}^G(i)})$. $\mathcal{D}_n$ denotes n iid samples from $\mathbb{P}$. A **multi-env CBN model** is tuples $M = (G^i, \mathbb{P}^i)_{i\in[I]}$ where each tuple is called an **environment (env)**, and each $\mathbb{P}^i$ is Markov relative to G^i. $\mathcal{D}_n^i$ denotes n iid samples from $\mathbb{P}^i$. Denote $\mathcal{M}$ as a (multi-env) **CBN model class** that contains different (multi-env) CBN models.

Definition 1 (Identifiability in causal discovery) *Given an observational CBN model class $\mathcal{M}$ in which all the joint distributions are absolutely continuous w.r.t. a measure μ, we say $\mathcal{M}$ is* ***identifiable*** *if for any $(G, \mathbb{P}_\theta), (G', \mathbb{P}_{\theta'}) \in \mathcal{M}$ with $\mathbb{P}_\theta(x) = \mathbb{P}_{\theta'}(x)$ μ-almost everywhere, we have $G = G'$. Given an multi-env CBN model class $\mathcal{M}$, we say $\mathcal{M}$ is* ***identifiable*** *if for any $(G^i, \mathbb{P}^i_\theta)_{i\in[I]}, (G'^i, \mathbb{P}^i_{\theta'})_{i\in[I]} \in \mathcal{M}$ with $\mathbb{P}^i_\theta(x) = \mathbb{P}^i_{\theta'}(x)$ μ-almost everywhere for all $i \in [I]$, we have $G^i = G'^i$ for all $i \in [I]$.*

For both observational and multi-env models, we have the following well-known result, similar to that in classical statistics (e.g. Greene, 2003, Thm 17.3). Defining the likelihood function $L(\theta|\mathcal{D}_n) = p_\theta(\mathcal{D}_n)$, we have the following result whose proof is deferred to § C.

Lemma 2 ***(Identifiability implies uniqueness of solution of minimum cross-entropy)***[2]

Given a CBN model class $\mathcal{M}$, if $\mathcal{M}$ is identifiable and its distribution class is parametric, then

(1) For the observational CBN model, the solution of maximum likelihood $\arg\max_{M\in\mathcal{M}} \lim_{n\to\infty} \frac{1}{n}\log L(M|\mathcal{D}_n)$ is unique. Equivalently, the minimizer of cross-entropy $\arg\min_{(G,\theta)\in\mathcal{M}} \mathbb{E}_{\mathbb{P}^*_\theta}[-\log \mathbb{P}_\theta(X)]$ is unique.

(2) For the multi-env CBN model with uniform prior over environments, the solution of maximum likelihood $\arg\max_{M\in\mathcal{M}} \lim_{n\to\infty} \sum_{i=1}^{K} \frac{1}{n}\log L(M|\mathcal{D}^i_n)$ is unique. Equivalently, there is a unique minimizer of the sum of cross-entropies across multi-env given by

$$\underset{\left((G_i)_{i\in[I]},(\theta_i)_{i\in[I]}\right)\in\mathcal{M}}{\arg\min} \sum_{i=1}^{I} \mathbb{E}_{\mathbb{P}_{\theta^*_i}}[-\log \mathbb{P}_{\theta_i}(X)]. \tag{2.1}$$

By Shannon's source coding theorem (Theorem 35) the entropy, (equivalently the minimum cross-entropy) is the shortest (most compressed) average coding length for an iid sequence. Therefore the desideratum of identifiability research is to justify that compression (minimum cross-entropy) is the correct model selection method in causal discovery.

Limitations

- **Identifiability is deterministic model selection.** By Lemma 2, once we have identifiability results in a model class $\mathcal{M}$, we can deterministically select the ground truth model in $\mathcal{M}$ using maximum likelihood or minimum cross-entropy. It is known that without constraints of distribution class and without knowledge of the intervention targets, there is no identifiability beyond the Markov equivalence class. There is extensive research on finding model classes that make causal models identifiable. These correspond to hard priors, restricting the model class to a lower dimensional submanifold, resulting in subjective model pre-selection. However, the model selection problem is inescapable —one can either pre-select the model class with constraints and then derive a deterministic model selection result, or perform model selection directly in an unconstrained model class. The success of modern empirical machine learning is based on the latter, while identifiability, based on the former, excludes parts of the model space a priori.
- **Intervention types and targets.** Many papers show identifiability results under no distribution constraints on causal mechanisms but with the knowledge of intervention types or targets. In Definition 1, we can see that intervention types or targets are constraints in $\mathcal{G}^I$, the I-th cartesian product of all graphs of d nodes. For example, the assumption of "all interventions are soft" (Perry et al., 2022; Wildberger et al., 2023) implies that the multi-env graphs $(G_i)_{i\in[I]}$ have to be the same graph G across all environments. Under faithfulness assumption, $\mathcal{M} = \bigsqcup_{(G_i)_{i\in[I]}\in\mathcal{G}^I} \Theta_{(G_i)_{i\in[I]}}$, a disjoint union of model classes, each of which contains I-env systems that are Markov and faithful to I-many graphs. By reducing the possible

[2] For readability we stay in the unconfounded setting and the strong version of identifiability. We can readily generalize Definition 1 and this lemma to identifiability up to an equivalence class, or generalize to the setting of causal representation learning, by changing the observed distribution to be a marginal distribution $\mathbb{P}_X$ of the model distribution $\mathbb{P}_{XZ}$, while forcing the invariance of $\mathbb{P}_{X|Z}$.

graphs, they in fact force hard priors over the probabilistic model class on multi-env systems. The more assumptions we have on intervention types or targets, the smaller the distribution class is, and the more chance of model misspecification there is. Without intervention types or targets, multi-env data are in fact *correlational* data, because every new environment can arise from interventions in each variable respectively, which can be completely unrelated to the mechanisms in the observational environment. In such cases, no identifiability beyond Markov equivalence class is possible.

- **Entropy is not all the bits needed to encode datasets.** The length of the codebook is missing in cross-entropy, which is why identifiability theories cannot distinguish different computational models that compute the same probabilistic model. We discuss this in detail in § B.3.

The central questions in this paper:

1. If we observe multi-env data, but have no certain knowledge about intervention types or targets, no hard prior over the model class, and thus no identifiability guarantees, can we decide on the causal relationship between variables, with a weaker definition of causality?
2. How to learn such a causal relationship? What is its relation with compression?

3. Algorithmic causality

We will now present an approach to describe causal models as probabilistic models implemented by Turing machines (TM). While this idea was first proposed by Janzing and Schölkopf (2010) for deciding the causal graphs among strings, our approach differs in that (1) our complexity score is fundamentally different from their Postulate 7; (2) we focus on the complexity of probabilistic models implemented by a specific class of Turing machines, see § 7 and § H for details.

	Identifiable, Pearl's causality	Algorithmic causality
Model class	A class of CBNs/SCMs which computes not all possible (multi-env) joint distributions	A class of CFMPs (Definition 9)/TMs which computes all possible (multi-env) joint distributions
Goal of model selection	Recover the ground truth graph up to certain symmetries by minimum cross-entropy (Lemma 2)	Ground truth is ill-defined. Just select the model that minimizes Kolmogorov/ FC complexity (Definition 14) among CFMPs
Is there model pre-selection before training	Yes. Hard priors (including intervention targets) are needed on probabilistic model classes; otherwise no identifiability beyond Markov equivalence class, even if we have multi-env datasets.	No. Constraining the class of CFMPs does not necessarily constrain the class of probabilistic models that those CFMPs can compute.
Subjectivity of model selection	The probabilistic model class where the groud truth is believed to live	Choice of UFCC (Definition 13) on which FC complexity is based

Table 1: Comparison between the learning of identifiable causal models and our algorithmic causal models.

Definition 3 (informal, algorithmic causality) *Consider a class of Turing machines where each T halts on any input in $\mathcal{X}^d := (\mathcal{B}^m)^d$ (see Definition 5) and outputs a codeword for x or outputs $\mathbb{P}(x)$ for a certain distribution $\mathbb{P}$. Suppose all T can simulate certain subprograms in the form of "If $X_i = \ldots$ then $X_j = \ldots$". We say that according to a model selection method (e.g. compression), X_i* ***algorithmically causes*** *X_j* ***locally*** *at $x \in \mathcal{X}^d$ if the method selects a Turing machine T such that, given the input x, among all the subprograms that T simulates there exists one in the form of "If $X_i = \ldots$ then $X_j = \ldots$" and there does not exist the opposite. If such a statement holds for all $x \in \mathcal{X}^d$, we say X_i* ***algorithmically causes*** *X_j.*

We emphasize that algorithmic causality is a property of the selected Turing machine w.r.t. a model selection method, not a property of the data distribution computed by a Turing machine. Notice that in identifiability research, the decision on causal graphs also depends on the model selection method (e.g. choice of the model class). In the following, we are interested in those model selection methods that do not constrain the joint distribution class.

A formal definition for a special case is provided below (Definition 38). We will see in § 4 that compression prefers selecting a Turing machine that says "X_i causes X_j" if the estimated conditional probability $\mathbb{P}(X_j|X_i)$ is approximately invariant across environments. Our model selection strategy is to constrain the class of Turing machines but not the distribution class that our Turing machine class can compute.

We consider discrete random variables, but it is a standard fact that they can approximate all absolutely continuous distributions with compact support to arbitrary precision:

Definition 4 *A* ***discrete distribution*** *$\mathbb{P}$ supported in the rational number space $(\mathbb{Q} \cap [0,1))^D$ is a function $(\mathcal{B}^*)^D \to \mathcal{B}^*$, where $\mathcal{B} = \{0,1\}$, and $\mathcal{B}^*$ is the set of binary sequences of arbitrary finite length, which is isomorphic to $\mathbb{Q} \cap [0,1)$ by the canonic dyadic expansion: $b_1 b_2 \cdots \in \mathcal{B}^*$ is equivalent to $\sum_{i=1}^{\infty} \frac{b_i}{2^i}$.[3] The (m,n)-**projection** $\mathbb{P}^{(m,n)} : (\mathcal{B}^m)^D \to \mathcal{B}^n$ is defined by*

$$\mathbb{P}^{(m,n)}(x) = \mathbb{P}(x|_m)|_n \tag{3.1}$$

where $x|_m$ denotes the m-length prefix of x. We call m the ***precision*** *of the variables in $\mathbb{P}$, and n the precision of the probability values in $\mathbb{P}$.*

We leave the more formal definition of above, using the notion of the **cylinder**, to § B.2.

Definition 5 *Given the precision m, we denote $\mathcal{X} := \mathcal{B}^m$. In this paper, we focus on probability distributions on the space $\mathcal{X}^d = (\mathcal{B}^m)^d$, where any $(x_1, \ldots x_d) \in \mathcal{X}^d$ is a concatenation of d-many binary sequences of length m. We call a* ***multi-env system*** *a list of discrete distributions $(\mathbb{P}^i)_{i \in [I]}$ on $\mathcal{X}^d$. I denotes the number of environments.[4] Equivalently, we can also write a multi-env system on $\mathcal{X}^{d-1}$ with less than 2^m environments as a discrete distribution on $\mathcal{X}^d$.*

Definition 6 *A Turing machine T is said to* ***compute*** *a function $f : A \subset \mathcal{X} \to \mathcal{B}^*$, if T rejects any input outside A and for all $x \in A$, $T(x) = f(x)$. We define the* ***equivalence relation*** *$\sim$ between two Turing machines S and T if they compute the same function.*

[3] For simplicity of the reading, in this paper, we do not allow deterministic distributions, i.e., there exists $x \in (\mathcal{B}^*)^D$ such that $\mathbb{P}(x) = 1$. In fact, to represent all values in $\mathbb{Q} \cap [0,1]$ with precision n with the value 1 included, $n+1$ bits are necessary instead of n bits.

[4] Below, we use the terms "distribution" and "multi-env system" interchangeably, since the latter can be written as a distribution over $\mathcal{X} \times [I]$. We are not given intervention targets, hence any variable can be the environment label.

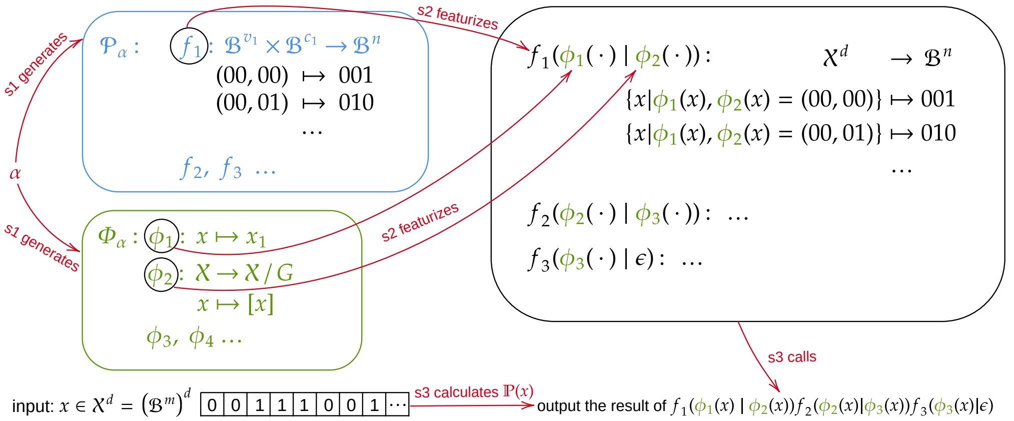
s1 generates
s1 generates
α
$\mathcal{P}_\alpha$:
f_1: $\mathcal{B}^{v_1} \times \mathcal{B}^{c_1} \to \mathcal{B}^n$
$(00,00) \mapsto 001$
$(00,01) \mapsto 010$
…
f_2, f_3 …
Φ_α : ϕ_1: $x \mapsto x_1$
ϕ_2: $\mathcal{X} \to \mathcal{X}/G$
$x \mapsto [x]$
ϕ_3, ϕ_4 …
s2 featurizes
s2 featurizes
$f_1(\phi_1(\cdot) \mid \phi_2(\cdot))$: $\mathcal{X}^d \to \mathcal{B}^n$
$\{x|\phi_1(x), \phi_2(x) = (00,00)\} \mapsto 001$
$\{x|\phi_1(x), \phi_2(x) = (00,01)\} \mapsto 010$
…
$f_2(\phi_2(\cdot) \mid \phi_3(\cdot))$: …
$f_3(\phi_3(\cdot) \mid \epsilon)$: …
s3 calls
input: $x \in \mathcal{X}^d = (\mathcal{B}^m)^d$
0 0 1 1 1 0 0 1 ⋯
s3 calculates $\mathbb{P}(x)$
output the result of $f_1(\phi_1(x) \mid \phi_2(x)) f_2(\phi_2(x)|\phi_3(x)) f_3(\phi_3(x)|\epsilon)$

Figure 1: Illustration of a CFMP (Definition 9). A CFMP α is a Turing machine that sequentially proceeds in three steps in red given any input in $\mathcal{X}^d$. Probabilistic mechanisms are blue and feature mechanisms are green. ϵ denotes the empty string. Before reading the input tape, α proceeds in two steps: generates $\mathcal{P}_\alpha$, Φ_α, and featurizes the probabilistic mechanisms. In the third step, α multiplies the conditional probabilities it needs for calculating $\mathbb{P}(x)$ and marginalizes over latent variables if there are hidden-variable mechanisms. We emphasize that this figure, or CFMP, is not a process of *learning*, but just a *model* in the model class where we proceed model selection.

We provide a more detailed introduction on computation theory in § B.1, and a more formal version of Definition 6 in Definition 22. We now introduce algorithmic models that can compute certain probability distributions. For a review of related approaches we refer to Section 7.

Definition 7 *A* ***probabilistic mechanism*** *is a Turing machine that computes the following discrete functions, which we call* ***probabilistic maps****: $f : \mathcal{B}^{v_f} \times \mathcal{B}^{c_f} \to \mathcal{B}^n$. We call v_f the bit length for the* ***value variable*** *of f, c_f the bit length for the* ***conditional variable*** *of f. Value and conditional variables are just the placeholders in the input of the probabilistic mechanism. The output $b_1 \dots b_n \in \mathcal{B}^n$ is equivalent to a rational number in $[0,1)$, i.e. $\sum_{i=1}^{n} \frac{b_i}{2^i}$. For a set of probabilistic mechanisms $\mathcal{P}$, we define its corresponding* ***probabilistic map set*** *$\overline{\mathcal{P}} := \mathcal{P}/\sim$.*

The key idea is that we do not sample from a distribution stochastically; we consider a probabilistic map as a deterministic function that maps points to values in $[0,1)$. A probabilistic mechanism is a Turing machine that computes such a function. In this paper, a **mechanism** always denotes a Turing machine instead of a function or distribution.

Definition 8 *Given the dimension d and precision m, we define the* ***feature mechanism set*** *Φ as a set of Turing machines that compute functions $\mathcal{X}^d \to \mathcal{B}^*$, which we call* ***feature maps****. We define its corresponding* ***feature map set*** *$\overline{\Phi} := \Phi/\sim$.*

In this paper, we focus on one class of Turing machines that we term CFMPs, and we show in § 5 that it aligns compression with causality and symmetry.

Definition 9 *For any discrete distribution $\mathbb{P}$, a* ***conditional feature-mechanism program (CFMP)*** *α is a Turing machine that computes $\mathbb{P}$ by doing the following steps (Figure 1):*

1. *Without reading the input tape, α generates $\mathcal{P}_\alpha$ and Φ_α in the memory, where Φ_α is a feature mechanism set, and $\mathcal{P}_\alpha$ is a probabilistic mechanism set.*

2. *Without reading the input tape, α featurizes the probabilistic mechanisms: for certain $f \in \mathcal{P}_\alpha$, α selects*[5] *certain $\phi : \mathcal{X}^d \to \mathcal{B}^{v_f}$ and $\psi : \mathcal{X}^d \to \mathcal{B}^{c_f}$ in Φ_α, and stores either the Turing machine which computes $x \mapsto f(\phi(x)|\psi(x))$ (termed "no-hidden-variable mechanism") or $(x, x') \mapsto f(\phi(x, x')|\psi(x, x'))$ (termed "hidden-variable mechanism"). Namely, while f maps $\mathcal{B}^{v_f} \times \mathcal{B}^{c_f} \to \mathcal{B}^n$, the featurized mechanism f_i maps $\mathcal{X}^d \to \mathcal{B}^n$ or $\mathcal{X}^d \times \mathcal{X}^d \to \mathcal{B}^n$.*
3. *The set of featurized mechanisms is written in the memory. For any input $x \in \mathcal{X}^d$, α selects*[6] *and computes the multiplication of a sequence of featurized mechanisms evaluated on x:*
   (a) *if only no-hidden-variable mechanisms are selected for x:*

   $$f_1(\phi_1(x)|\psi_1(x)) \dots f_k(\phi_k(x)|\psi_k(x))$$

   for certain k, then compute and output the result.

   (b) *if there exist hidden-variable mechanisms in the formula*

   $$f_1(\phi_1(x, x')|\psi_1(x, x')) \dots f_k(\phi_k(x, x')|\psi_k(x, x'))$$

   (where x' is a formal placeholder in the formula without value assignment) for certain k, then marginalize over $x' \in \mathcal{X}^d$ and output the result.

Example 1 *The following examples (with more in Example 4 and Example 5) show that the class of CFMPs can model a wide range of probabilistic models, which turn into classes of Turing machines instead of distributions. All of the examples except causal representation learning models use only no-hidden-variable feature mechanisms. In the following examples, $\mathbb{P}$ denotes respectively a discrete (multi-env) system that the CFMP computes:*

1. *A* ***causal Bayesian network (CBN)*** *model computing $\mathbb{P}$ on $\mathcal{X}^d$ is a CFMP α such that*
   (a) *the feature map set is $\overline{\Phi_\alpha} \subset \{(\cdot)_A | A \subset 2^{[d]}\}$, projections over coordinates.*
   (b) *In step 3, for all $x \in \mathcal{X}^d$, α computes $\mathbb{P}(x)$ using the same set of featurized mechanisms: $f_1(x_{A_1}|x_{B_1}) \dots f_k(x_{A_k}|x_{B_k})$, such that $\bigsqcup_{i=1}^k A_i = [d]$ and $A_i \cap B_i = \emptyset \quad \forall i$ and $B_i \subset \cup_{j=1}^{i-1} A_j \quad \forall i$.*
2. *A* ***causal representation learning*** *model computing $\mathbb{P}$ on $\mathcal{X}^d$ is a CFMP α such that*
   (a) *in $\mathcal{P}_\alpha$ there exists a probabilistic mechanism $f : \mathcal{X}^d \times \mathcal{Z}^d \to \mathcal{B}^n$, where $\mathcal{Z}^k := (\mathcal{B}^{m'})^k$ denotes a space for hidden variables, of arbitrary precision m' and dimension k.*
   (b) *There exists a feature mechanism $\psi : \mathcal{X}^d \to \mathcal{Z}^k$ that featurizes f into a hidden-variable mechanism $(x, x') \mapsto f(x|\psi(x')) : \mathcal{X}^d \times \mathcal{X}^d \to \mathcal{B}^n$.*
   (c) *In step 3, for all input $x \in \mathcal{X}^d$, α outputs $\sum_{x' \in \mathcal{X}^d} f(x|\psi(x'))g(\psi(x'))$, where g is any featurized mechanism, or a product of different mechanisms.*
3. *A G-****invariant learning*** *model computing $\mathbb{P}$ on $\mathcal{X}^2$ is a CFMP α where G is a group that acts on $\mathcal{X}$ and*[7]

[5] This selection procedure determines a function: {featurized mechanisms generated in Step 2} → $\mathcal{P}_\alpha$. Note that the bit length is not intrinsic to the function, but depending on the CFMP that computes this function.

[6] In principle the selection of featurized mechanisms in Step 3 depends on $x \in \mathcal{X}^d$. Namely, this selection procedure determines a function $\mathcal{X}^d \to$ {featurized mechanisms generated in Step 2}.

[7] For simplicity of presentation, we give examples that compute one-dimensional invariant and equivariant models. One can easily generalize examples 4, 5 and 6 to higher dimensions.

(a) *there exists $\phi : \mathcal{X} \to \mathcal{X}/G$ in Φ_α partitioning the orbits of $\mathcal{X}$ under the action of G. In addition, Φ_α contains projections on two coordinates π_1 and π_2. Φ_α also contains $\phi \circ \pi_1$. There exists $f_1, f_2 \in \mathcal{P}_\alpha$ such that $f_1(\pi_2(x)|\phi \circ \pi_1(x)) = \mathbb{P}(x_2|\phi(x_1))$, and $f_2(\pi_1(x)) = \mathbb{P}(x_1)$ for all $x \in \mathcal{X}^2$.*

(b) *In step 2, α featurizes the probabilistic mechanisms: $f_1 \mapsto f_1(\pi_2(\cdot)|\phi \circ \pi_1(\cdot))$, $f_2 \mapsto f_2(\pi_1(\cdot))$, and $f_3 \mapsto f_3(\pi_1(\cdot))$.*

(c) *In step 3, for all $x \in \mathcal{X}^2$, α computes $\mathbb{P}(x) = \mathbb{P}(x_2|\phi(x_1))\mathbb{P}(x_1) = f_1(\pi_2(x)|\phi \circ \pi_1(x))f_2(\pi_1(x))$.*

4. *A **statistical density estimator** computing $\mathbb{P}$ on $\mathcal{X}^d$ is a CFMP α that*

   (a) *in $\mathcal{P}_\alpha$ there is only one Turing machine, which computes the distribution $\mathbb{P} : \mathcal{X}^d \to \mathcal{B}^n$.*

   (b) *In Φ_α there is only one Turing machine, which computes identity Id : $\mathcal{X}^d \to \mathcal{X}^d$.*

   (c) *In step 2, P is featurized trivially by identity feature map, so $\mathbb{P}$ is not changed.*

   (d) *In step 3, for all $x \in \mathcal{X}^d$, α outputs $\mathbb{P}(x)$.*

Now we concretize the general definition of algorithmic causality (Definition 3) for a class of Turing machines, i.e., CFMPs. First, let us focus on the class of CFMPs that compute only CBNs. This is the most familiar case to the readers in causality.

Definition 10 (Algorithmic causality in CBN) *Given a class of CFMPs that compute the causal Bayesian network models, for $A, B \subseteq [d]$ with $A \cap B = \emptyset$, we say that according to a model selection method[8], X_A **algorithmically causes** X_B **locally at** $x \in \mathcal{X}^d$ if the method selects a CFMP α such that, in the third step in α the featurized mechanisms for x include one mechanism $f(X_B|X_A)$ in which X_B is a set of value variables and X_A is a set of conditional variables (Definition 7), and do not include mechanisms with the opposite direction $f(X_A|X_B)$. Similarly, we say that X_A **algorithmically causes** X_B if the above assumption holds for all $x \in \mathcal{X}^d$.*

We observe that the causal variables X_i are in fact $\pi_i(X)$, the i-th projection over X. We generalize the definition above to the case of arbitrary feature map, in § D (Definition 38).

4. Learning algorithmic causality by compression

In § 3, we introduced a computational model that can compute a wide range of probabilistic models. Recall that in our problem setting, no identifiability beyond the Markov equivalence class is possible for probabilistic models. However, model selection beyond the Markov equivalence class is possible for our computational model, e.g., based on the coding length of the model. This model selection is crucial because it determines algorithmic causations (Definition 38). Since the goal of identifiability is to justify that the model that minimizes the multi-env cross-entropy should be the preferred model (in that case, the ground truth), we generalize that principle:

Principle 11 *Given (multi-env) datasets, the preferred (causal) models to select (no matter algorithmic or probabilistic) are the ones that minimize the bit length of a sender's message enabling the receiver to reconstruct the multi-env datasets.*

[8] Such as minimizing the finite codebook complexity, which we introduce in § 4 and Definition 14.

Notice that this principle holds in classical causal discovery whether the model class is identifiable or not because the model that minimizes cross-entropy also maximizes likelihood, which is preferred over those that are less likely. As discussed in § B.3, entropy is the minimal average data-to-model coding length *given a codebook*, therefore it is not the *overall* bit length needed by a sender. Algorithmic causality makes the codebook length unignorable because the length of CFMP is upper boundable (sometimes computable) and unignorable when the data is finite. The rigorous *overall* bit length that a sender needs for lossless encoding of a data sequence x is called *Kolmogorov complexity* $C(x)$. We build upon background knowledge of Kolmogorov complexity as summarized in Appendix § E.1. In the following, we give an upper bound of Kolmogorov complexity and the regularity part $l_U(T)$ in the discrete finite sample space.

After decomposing the Kolmogorov complexity of a multi-env dataset $C_U(x_1, \dots x_n)$ into a two-part code $l_U(T) + C_U(x_1, \dots x_n | T)$ (Lemma 42), there is still an uncomputable part $l_U(T)$. We can construct an upper bound of $l_U(T)$ by constraining the Turing machine class, without constraining the distribution class or codebook class that our Turing machines can compute.

Definition 12 *We say that an injective function $g : \mathcal{A} \subset \mathcal{X}^d \to \mathcal{B}^*$ is a* ***finite codebook*** *if the set $g(\mathcal{A})$ is prefix-free. We say that a Turing machine T is a* ***finite coding mechanism (FCM)*** *if it computes a finite codebook.*

Definition 13 *Given the dimension d and precision m of $\mathcal{X}^d$, we say that a Turing machine V is a* ***universal finite codebook computer (UFCC)***[9] *if*

1. *V takes input $\langle k, p \rangle$, where k is the index of an FCM in a decidable set*[10] *of FCMs; p is a natural number, which is equivalent to a binary string $B(p)$ (see Definition 19); same as Lemma 42, $\langle \cdot, \cdot \rangle$ is the self-delimiting concatenation.*

2. *for any finite codebook $g : \mathcal{A} \subset \mathcal{X}^d \to \mathcal{B}^*$, there exists k such that $V(\langle k, \cdot \rangle)$ is an FCM computing $(g^*)^{-1}$, which is a partial (i.e., not everywhere defined) function $\mathcal{B}^* \to \mathcal{A}^*$, and g^* is the extension of the codebook g (see Definition 30, $(g^*)^{-1}$ is well-defined because g is prefix-free);*

3. *$V(\langle k, p \rangle)$ is computed by decoding the binary string $B(p)$ using $V(\langle k, \cdot \rangle)$.*

Definition 14 *Given a UFCC V, for any finite codebook g, the* ***finite codebook (FC) complexity*** *is defined as*

$$C_V^{FC}(x_1 \dots x_n) := \min_{(k,p) \in \mathbb{N}^2} \{ l(\langle k, p \rangle) : V(\langle k, p \rangle) = x_1 \dots x_n \}. \tag{4.1}$$

We now first apply Lemma 40 to upper bound $C_U(x_1 \dots x_n)$ by $C_V^{FC}(x_1 \dots x_n)$, and then split $C_V^{FC}(x_1 \dots x_n)$ into a two-part code using the idea similar to Lemma 42.

For any additively optimal (Lemma 40) UTM U and any UFCC V, using the same argument as in the proof of Lemma 40, there exists n such that $V = T_n$, so we can bound the Kolmogorov complexity C_U by FC complexity C_V^{FC}:

$$C_U(x_1 \dots x_n) \le C_V^{FC}(x_1 \dots x_n) + O(1). \tag{4.2}$$

[9] For simplicity of presentation we do not set (m, d) as part of the input in a UFCC, while in the general sense, a UFCC should take m, d, k, p as input. Suppose we are given a recursively enumerable set of FCMs for each (m, d), then we can construct a UFCC inputting the self-delimited (m, d, k, p), i.e. $\langle m, \langle d, \langle k, p \rangle \rangle \rangle$.

[10] Namely, the set is recursively enumerable (r.e.) and its complement set in the set of all Turing machines is also r.e.

Replace C_V^{FC} by Definition 14,

$$C_U(x_1 \dots x_n) \leq \min_{k,p}\{l(\langle k,p \rangle) : V(\langle k,p \rangle) = x_1 \dots x_n\} + O(1) \tag{4.3}$$

$$= \min_{T,p}\{2l_V(T) + l(p) : T(p) = x_1 \dots x_n\} + O(1) \tag{4.4}$$

where $2l_V(T)(+1)$ is the self-delimiting code length (Definition 20) of the FCM T in the effective enumeration (Definition 39) of FCMs according to V, and $l(p)$ is the literal length of the binary code that T takes as input and thereby outputs $x_1 \dots x_n$.

Given a UFCC V, given any finite codebook $g : \mathcal{X}^d \to \mathcal{B}^*$ and a number p such that $(g^*)^{-1} \circ B(p) = x_1 \dots x_n$, there exists a FCM T that computes $(g^*)^{-1}$, and thus $C_U(x_1 \dots x_n)$ is upper bounded by $2l_V(T) + l(p)$. In particular, if we believe that the data $x_1 \dots x_n$ is iid sampled from a (multi-env) system or discrete distribution, then we should use a Shannon codebook (i.e. the codeword length for x is $-\log \mathbb{P}_{\hat{\theta}}(x)$) and a Shannon codeword p (i.e. $p = g(x_1) \dots g(x_n)$), where $\hat{\theta}$ is the maximum likelihood estimator in the class of all the discrete distributions supported in $\mathcal{X}^d$. This is justified by Shannon's source coding theorem: a shortest p when $n \to \infty$ is $g(x_1) \dots g(x_n)$.

There is a trade-off between $l_V(T)$ and $l(p)$: if $n \to \infty$ then $l(p) \to nH(\mathbb{P}_{\hat{\theta}})$ dominates. In the finite data case $l_V(T)$ (codebook part) is not negligible.

Definition 15 *Given $\mathcal{A} \subset \mathcal{X}^d$, We say that a Turing machine γ is a **Huffman coding program** if given any discrete finite distribution $\mathbb{P}$ supported on $\mathcal{A}$ as input it outputs a Turing machine that computes a Huffman code (Definition 36) of $\mathbb{P}$.*

One example of UFCC is: a Turing machine $V_{\mathcal{C}}$ that takes any element in a finite class $\mathcal{C}$ of CFMP as input and combines it with a Huffman coding program to compute a Huffman code, then decodes an integer p which is equivalent to an input binary sequence $B(p)$. More examples are in § E.2. We also compare different choices of UFCC in § E.3.

Given precision (m, n) for $\mathcal{X}^d$, any UFCC can simulate any finite codebook of this precision. By Corollary 34, each prefix code corresponds to a unique probability semi-measure, therefore any UFCC can express any probability measures in $\mathcal{X}^d$ in precision (m, n).

Given the precision (m, n) and dimension d, given a multi-env dataset on $\mathcal{X}^d$, we can minimize the upper bound of FC complexity eq. (4.4) in a class of Turing machines or CFMPs. For many UFCCs, computing FC complexity is very hard. By Corollary 34,there is a one-to-one correspondence between code length functions and probability distributions. For those UFCCs that simulate a CFMP followed by a *fixed* Huffman coding program, it suffices to calculate the bit length needed in the CFMP part in order to perform model selection. In § 5 we compare the bits needed by different CFMPs under a given UFCC.

5. Case studies

We study the solutions for eq. (4.4) under some particular UFCCs, showing that compression leads to selecting CFMPs that have algorithmic causal or symmetric structures. In the following, all the UFCCs simulate FCMs by composing a CFMP with a Huffman coding program, and all the CFMPs only involve no-hidden-variable mechanisms.

5.1. Causal factorizations and sparse mechanism shifts

Consider a UFCC U_{TabCBN} that first takes m, d, n as input where m denotes the precision of $\mathcal{X}$, i.e., $|\mathcal{X}| = 2^m$, and d denotes the number of variables, and n denotes the precision of discrete distribution values. Let U_{TabCBN} only allow the CFMPs in the following form: each CFMP α is a causal Bayesian network defined in 1. of Example 1, with the further constraints that

1. All elements in $\mathcal{P}_\alpha$ are conditional probability tables. Each $f \in \mathcal{P}_\alpha$ is incompressible, i.e., the Kolmogorov complexity $C(f)$ equals the coding length of its probability table.
2. Φ_α is restricted to projections on variables: for d variables, there are 2^d possible projections, and we define a uniform code on these elements so that each element needs d bits.

Proposition 16 *(Causal factorization is shorter than encoding the joint distribution) Suppose $\mathbb{P}$ is supported on $\mathcal{X}^d \times [I]$ with precision (m, n) and $\mathbb{P}(X, e_i) = \mathbb{P}^i(X_1|X_{S_i}, e_i)\mathbb{P}(X_2, \ldots, X_d)\mathbb{P}(e_i)$ (w.l.o.g. suppose $\mathbb{P}(e_i) = \frac{1}{I}$) and $(S_i)_{i\in[I]}$ are subsets of $[d]$ with $|S_i| < d-1$. Suppose $I = 2^{\log I} \leq 2^n$. Denote α, β as two CFMPs simulated by U_{TabCBN} and computing the same $\mathbb{P}$, while α is a CBN model (Example 1 (1)) proceeds by separately saving and calling factorized mechanisms, and β is a statistical density estimator (Example 1 (4)) which proceeds by encoding the whole multi-env distribution in $\mathcal{P}_\beta$. Then $l_{U_{TabCBN}}(\alpha)(m, d) = o(l_{U_{TabCBN}}(\beta)(m, d))$ as $m \to \infty$ or $d \to \infty$.*

Although U_{TabCBN} can compute any finite codebook, and by Corollary 34, it can equivalently compute any multi-env distribution (Corollary 34), its encoding is inefficient because it encodes all elements in $\mathcal{P}_\alpha$ in the form of probability tables. To relax this assumption, we need to assume compressibility of $\mathcal{P}_\alpha$. We explain intuitively Proposition 46, which can be found in § F.3:

Proposition 17 *(Informal explanation of Proposition 46) In a certain UFCC $U_{compCBN}$ defined before Proposition 46, consider a set of CFMPs simulated by $U_{compCBN}$ that compute the same multi-env distribution. Suppose in step 2 of those CFMPs, they all generate M-many featurized mechanisms, and suppose in step 3, N-many featurized mechanisms are written in the memory with repetitions allowed; then, when the actual (non-repetitive) number of needed mechanisms k is small, those CFMPs that first choose k mechanisms from step 2 and then use them to instantiate N-many mechanisms in step 3 is of shorter coding length than those directly assign M to N-many mechanisms.*

5.2. Symmetries

Consider a UFCC U_{TabInv} that first takes m, d, n as input where m denotes the precision of $\mathcal{X}$, i.e., $|\mathcal{X}| = 2^m$, and d denotes the number of variables, and n denotes the precision of discrete distribution values. Let U_{TabInv} only allow the CFMPs in the form of (4) in Example 1 and with further constraint same as the first constraint in the definition of U_{TabCBN}, plus that

1. For all α simulated by U_{TabInv}, Φ_α is restricted to the compositions of projections on variables and quotient maps $\phi_G : \mathcal{X} \to \mathcal{X}/G$ of a certain group action G that only acts on one dimension in $\mathcal{X}^d$ and leaves other dimensions unchanged. Namely, any element in Φ_α is in the form $\phi_G \circ \pi$.[11]
2. The selected mechanisms in Step 3 is the same for all $x \in \mathcal{X}^d$, namely there is no context-specific mechanism (Example 1 (2)).

[11] If we allow more flexible quotient maps we can detect more symmetric structures. Those constraints are written in the UFCC, which is shared by the sender and receiver, so they do not take up bits in the FC complexity.

Proposition 18 *(Invariance factorization is shorter than Markov factorization) Suppose $\mathbb{P}$ is supported on $\mathcal{X}^2$ and $\mathbb{P}(X) = \mathbb{P}(X_1|\phi(X_2))\mathbb{P}(X_2)$, where $\phi : \mathcal{X} \to \mathcal{X}/G$ is the quotient map of a certain group action G that only acts on one dimension in $\mathcal{X}^2$ and leaves the other dimension unchanged. Suppose the number of orbits $\mathcal{X}/G$ is independent of the precision m. Denote α, β as two CFMPs simulated by U_{TabInv} and computing the same $\mathbb{P}$, while α is a G-invariant learning model (Example 1 (3)) which factorizes $\mathbb{P}$ into $\mathbb{P}(X) = f_1(\pi_1(X)|\phi \circ \pi_2(X))f_2(\pi_2(X))$, and β is a CBN model (Example 1 (1)) which factorizes $\mathbb{P}$ into $\mathbb{P}(X) = f_1'(\pi_1(X)|\pi_2(X))f_2(\pi_2(X))$ for certain f_1, f_1', f_2. Then $l_{U_{TabInv}}(\alpha)(m) = o(l_{U_{TabInv}}(\beta)(m))$.*

With a proof similar to Proposition 16 we can show that for the general case $\mathbb{P}(X) = \mathbb{P}(X_1|\phi \circ \pi(X))\mathbb{P}(\pi(X))$, if the dimension of $\pi(\mathcal{X}^d)$ is lower than $d-1$, then the model length of the Markov factorization β in Proposition 18 is shorter than the statistical density estimator. In this case, invariant factorization is shorter than Markov factorization, which is shorter than no factorization.

The orbits of many group actions are independent of the precision m, such as the reflection and translation; rotation is also the case when m is sufficiently large.

6. Experiments

We illustrate our theoretical findings through simple experiments with synthetic data[12]. The results are in Appendix § G. We consider two synthetic settings with sparse mechanism shifts. In both settings, the goal is to show that by minimizing FC complexity, we select a model that trades off the complexity of the model itself and the data-to-model coding length, i.e. negative log-likelihood. We also infer algorithmic causes according to the model selection method of minimizing FC complexity.

7. Related work

Janzing and Schölkopf (2010) were the first to consider causal mechanisms implemented by Turing machines. They replace statistical (conditional) independence with algorithmic (conditional) independence, conjecturing that if the Kolmogorov complexity of a string or a joint distribution can be decomposed into a sum of conditional Kolmogorov complexities of the causal mechanisms according to a graph, then this graph should be selected. They also extended their model selection principle to probabilistic models (Janzing and Schölkopf, 2010, Postulate 7), which we comment in detail in § H and compare with our Principle 11. The incomputable objective in (Janzing and Schölkopf, 2010, Postulate 7) is replaced by entropy (Steudel et al., 2010; Pranay and Nagaraj, 2021) or MDL (Budhathoki and Vreeken, 2016, 2017; Marx and Vreeken, 2019a; Mian et al., 2021, 2023) in all subsequent papers. We discuss in § B.3 and § I the difference between our approach and them. Marx and Vreeken (2021) discusses the relationship between Postulate 7 and two-part code. We give our comments on their results and on Postulate 7 in § H. Some papers (Marx and Vreeken, 2019b,c; Mameche et al., 2022, 2024) claim identifiability (i.e., recovery of the ground-truth graph) by minimizing their proxy of Postulate 7 in Janzing and Schölkopf (2010). We show in Lemma 2 that any identifiability of graphs is reduced to the claim of the uniqueness of the solution of minimum cross-entropy instead of minimizing a bound of Kolmogorov complexity. Our approach focuses on an upper bound of Kolmogorov complexity of a specific class of Turing machines that compute probabilistic models. Our objective is fundamentally different from (Janzing and Schölkopf, 2010, Postulate 7), as we show in § H.

[12] The code for the experiments can be found here: https://github.com/WendongL/algorithmic-causality-compression

Dhir et al. (2024) address bivariate causal discovery without confounding by comparing the posteriors of two graphs, with the correctness (probability of selecting the ground truth graph) depending on the total variation between the ground-truth distribution and marginal likelihood. In § I, we discuss the difference between our approach and Bayesian model selection.

On the side of computation theory, there is abundant work on constraining the definition of Kolmogorov complexity to make it computable, such as resource-bounded complexity (Barzdin, 1968), logical depth (Chaitin, 1977), automatic complexity (Shallit and Wang, 2001).Those definitions are fit for compressing more general strings without any probabilistic structure, therefore the entropic code ($-\log \mathbb{P}$) is not applicable. In the domain of knowledge representation, Shen et al. (2018); Kisa et al. (2014) use probabilistic sentential decision diagrams (PSDD) to model Bayesian networks and learn them by maximum likelihood. Some use instead trees (Chen and Darwiche, 2022) and arithmetic circuits (Darwiche, 2022; Huang and Darwiche, 2024).

8. Discussion

If it is the case that compression in some cases may *automatically* yield causal structure, then this has significant implications for modern machine learning. For instance, there is an ongoing debate as to whether large language models (LLMs) can understand causality in the sense of correctly applying causal principles across a range of problems (Jin et al., 2023). After all, a large extent of apparent causal knowledge may be explained by simply regurgitating causal knowledge which is abundant in the training set, and one may thus argue that the apparent causal knowledge of LLMs may be entirely superficial. The arguments put forward in the present paper suggest that perhaps these two extremes may not be entirely irreconcilable: training a (large, but finite) model on a significant fraction of the internet necessarily forces a model to compress data, and even though the classical identifiability assumptions do not hold, an algorithmic causal model can still be extracted. In other words, if the empirical scaling laws for LLMs continue to hold, the model may have no choice but to learn (algorithmic) causality. Similar indirect arguments have been made for non-causal learning for the case of grokking (Power et al., 2022) and the emergence of complex skills (Arora and Goyal, 2023). An open question is whether LLMs simulate nontrivial CFMPs, i.e., whether they call and reuse some mechanisms like in CFMP. We know that in principle, they can (Pérez et al., 2021).

The use cases of algorithmic causality and Pearl's causality are disjoint. Algorithmic causality is not a competitor of Pearl's causality, since (i) if we have infinite data and the likelihood converges, compression is trivialized to minimum cross-entropy, which outputs any Turing machine that computes the same probability distribution, so the model selection by any UFCC suffers from the same non-identifiability as Pearl's framework; (ii) algorithmic causality deals with multi-env but correlational data and is a rung 1 model in Pearl's hierarchy; (iii) its advantage lies only in those cases where the intervention targets are not certain, for example in LLM pre-training, there is no prior knowledge about which context token is the environment label or intervention-target variable. In this case, using Pearl's causality might not be as appropriate as using algorithmic causality; on the other hand, if scientists have data generated by some strictly randomized controlled experiments on many variables respectively, then the identifiability results in Pearl's causal models are more convincing than algorithmic causality for the scientific community.

Acknowledgments

The authors thank Frederik Hytting Jørgensen and the anonymous reviewers for helpful comments and discussions. This work was supported by the Tübingen AI Center.

References

Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models, 2023. arXiv:2307.15936.

Ya M Barzdin. Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. In *Soviet Mathematics Doklady*, volume 9, page 122, 1968.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123, 1996.

Kailash Budhathoki and Jilles Vreeken. Causal inference by compression. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 41–50, 2016. doi: 10.1109/ICDM.2016.0015.

Kailash Budhathoki and Jilles Vreeken. Mdl for causal inference on discrete data. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 751–756. IEEE, 2017.

Neal L Carothers. *Real analysis*. Cambridge University Press, 2000.

Gregory J Chaitin. Algorithmic information theory. *IBM journal of research and development*, 21 (4):350–359, 1977.

Yizuo Chen and Adnan Darwiche. On the definition and computation of causal treewidth. In *Uncertainty in Artificial Intelligence*, pages 368–377. PMLR, 2022.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Adnan Darwiche. Causal inference using tractable circuits. *arXiv preprint arXiv:2202.02891*, 2022.

Anish Dhir, Samuel Power, and Mark van der Wilk. Bivariate causal discovery using Bayesian model selection. In *Forty-first International Conference on Machine Learning*, 2024.

William H Greene. *Econometric analysis*. Pearson Education India, 2003.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Sharut Gupta, Stefanie Jegelka, David Lopez-Paz, and Kartik Ahuja. Context is environment. In *The Twelfth International Conference on Learning Representations*, 2023.

Fred C Hennie and Richard Edwin Stearns. Two-tape simulation of multitape turing machines. *Journal of the ACM (JACM)*, 13(4):533–546, 1966.

Haiying Huang and Adnan Darwiche. Causal unit selection using tractable arithmetic circuits. *arXiv preprint arXiv:2404.06681*, 2024.

Dominik Janzing and Bernhard Schölkopf. Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010.

Z. Jin, Y. Chen, F. Leeb, L. Gresele, O. Kamal, Z. Lyu, K. Blin, F. Gonzalez, M. Kleiman-Weiner, M. Sachan, and B. Schölkopf. Cladder: Assessing causal reasoning in language models. In *Advances in Neural Information Processing Systems 36*, volume 36, pages 31038–31065. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/631bb9434d718ea309af82566347d607-Paper-Conference.pdf.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

Andrei Nikolaevic Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2019.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

Sarah Mameche, David Kaltenpoth, and Jilles Vreeken. Discovering invariant and changing mechanisms from data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1242–1252, 2022.

Sarah Mameche, David Kaltenpoth, and Jilles Vreeken. Learning causal models under independent changes. *Advances in Neural Information Processing Systems*, 36, 2024.

Alexander Marx and Jilles Vreeken. Causal inference on multivariate and mixed-type data. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II 18*, pages 655–671. Springer, 2019a.

Alexander Marx and Jilles Vreeken. Identifiability of cause and effect using regularized regression. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 852–861, 2019b.

Alexander Marx and Jilles Vreeken. Telling cause from effect by local and global regression. *Knowledge and Information Systems*, 60:1277–1305, 2019c.

Alexander Marx and Jilles Vreeken. Formally justifying mdl-based inference of cause and effect. *arXiv preprint arXiv:2105.01902*, 2021.

Osman Mian, Michael Kamp, and Jilles Vreeken. Information-theoretic causal discovery and intervention detection over multiple environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9171–9179, 2023.

Osman A Mian, Alexander Marx, and Jilles Vreeken. Discovering fully oriented causal networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8975–8982, 2021.

Junhyung Park, Simon Buchholz, Bernhard Schölkopf, and Krikamol Muandet. A measure-theoretic axiomatisation of causality. *Advances in Neural Information Processing Systems*, 36: 28510–28540, 2023.

Judea Pearl. *Causality*. Cambridge University Press, 2009.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021.

Ronan Perry, Julius Von Kügelgen, and Bernhard Schölkopf. Causal discovery in heterogeneous environments under the sparse mechanism shift hypothesis. *Advances in Neural Information Processing Systems*, 35:10904–10917, 2022.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

SY Pranay and Nithin Nagaraj. Causal discovery using compression-complexity measures. *Journal of Biomedical Informatics*, 117:103724, 2021.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Jeffrey Shallit and Ming-Wei Wang. Automatic complexity of strings. *Journal of Automata, Languages and Combinatorics*, 6(4):537–554, 2001.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Yujia Shen, Arthur Choi, and Adnan Darwiche. Conditional psdds: Modeling and learning with modular knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

B. Steudel, D. Janzing, and B. Schölkopf. Causal Markov condition for submodular information measures. In A. Kalai and M. Mohri, editors, *Conference on Learning Theory (COLT)*, pages 464–476, Madison, WI, USA, 2010. OmniPress.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

Jonas Bernhard Wildberger, Siyuan Guo, Arnab Bhattacharyya, and Bernhard Schölkopf. On the interventional kullback-leibler divergence. In *Conference on Causal Learning and Reasoning*, pages 328–349. PMLR, 2023.

APPENDIX

Overview

- Appendix § A recapitulates the notation used in this paper.
- Appendix § B contains preliminary backgrounds.
  - Appendix § B.1 contains preliminaries in computation theory.
  - Appendix § B.2 contains preliminaries in discrete probability theory.
  - Appendix § B.3 contains preliminaries in compression (source coding) in information theory.
- Appendix § C contains the proof in § 2.
- Appendix § D contains the supplemental theory in § 3.
- Appendix § E contains the supplemental theory in § 4.
- Appendix § F contains supplemental results and proofs in § 5.
- Appendix § G contains details of experiments in § 6.
- Appendix § H contains remarks on the Algorithmic Markov Condition (Janzing and Schölkopf, 2010) and the subsequent works based on it.
- Appendix § I contains remarks on the Minimum Description Length (MDL) principle and Bayesian model selection, their difference and relation to our approach.

Appendix A. Notations

Symbol	Description
log	Simplified symbol for $\log_2$
$\mathbb{Q}$	The set of rational numbers
G	A directed acyclic graph with nodes $V = [d]$ and arrows E
$[d]$	The natural numbers $1, \ldots, d$
$\mathrm{pa}^G(i)$	Parents of i, defined as $\{j \in V(G) \mid (j, i) \in E(G)\}$
$\mathcal{B}$	Binary alphabet $\{0, 1\}$
$\mathcal{X}$	The discrete finite space for one dimension of samples, of cardinal 2^m
Ω	The discrete finite space for one dimension of the formal variable ω, of cardinal 2^m
ϵ	Empty string
α	A conditional feature-mechanism program (CFMP) (Definition 9)
Γ_x	Cylinder, defined in Definition 25
$\mathcal{P}_\alpha$	List of probabilistic mechanisms (Definition 7) generated by CFMP α
Φ_α	Set of feature mechanisms (Definition 8) generated by CFMP α
f_1	A probabilistic mechanism (Definition 7)
$g.x$	Group action on $x \in \mathcal{X}^d$, with $g \in G$ for a certain group G
$C_U(x)$	Kolmogorov complexity of a string x under a universal Turing machine U (Definition 39)
$\bar{n}$	Self-delimiting code of a natural number n, see Definition 20
$\langle x, y\rangle$	Self-delimiting concatenation of natural numbers x, y, defined in Definition 20
$l_U(T)$	Length of the index of Turing machine T in the effective enumeration of Turing machines in a universal Turing machine or UFCC U, see Definition 39
$l(n)$	Length of a binary string or equivalently a natural number n in the binary representation, defined in eq. (B.3)
U_{TabCBN}	A UFCC that is defined and used in § 5. Same for $U_{\text{CompCBN}}, U_{\text{TabInv}}$

Appendix B. Preliminaries

B.1. Computation theory

We introduce some notions that we mentioned in the paper. We follow Li et al. (2019).

We use an alphabet of binary symbols $\mathcal{B} = \{0, 1\}$. The set of all finite strings over $\mathcal{B}$ is denoted by $\mathcal{B}^*$, defined as

$$\mathcal{B}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\} \tag{B.1}$$

with ϵ denoting the empty string, with no letters. Concatenation is a binary operation on the elements of $\mathcal{B}^*$ that associates xy with each ordered pair of elements (x, y) in the Cartesian product $\mathcal{B}^* \times \mathcal{B}^*$.

We now consider a correspondence of finite binary strings and natural numbers. The standard binary representation has the disadvantage that either some strings do not represent a natural number, or each natural number is represented by more than one string. For example, either 010 does not represent 2, or both 010 and 10 represent 2. We can map $\mathcal{B}^*$ one-to-one onto the natural numbers by associating each string with its index in the length-increasing lexicographic ordering

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \ldots \tag{B.2}$$

Definition 19 *We call the* ***binary lexicographic code*** B *the map that maps eq.* (B.2) *reversely, i.e.* $0 \mapsto \epsilon$, $1 \mapsto 0$, $2 \mapsto 1$, $3 \mapsto 00 \dots$.

The length of a finite binary string x is the number of bits it contains and is denoted by $l(x)$. One can verify that for a natural number n represented by eq. (B.2),

$$l(n) = \lfloor \log_2(n+1) \rfloor \tag{B.3}$$

For the readability of the paper, we define $l(n) = \log_2 n$ instead.

Definition 20 *A* ***self-delimiting code*** *of a natural number* $n \in \mathbb{N}$ *is a function* $\mathbb{N} \to \mathcal{B}^*$ *that maps* $n \in \mathbb{N}$ *to*

$$\bar{n} = \underbrace{11 \dots 1}_{l(n) \text{ times}} \quad 0B(n). \tag{B.4}$$

where B *is binary lexicographic code defined in Definition 19.*

We call $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathcal{B}^*$ *a* ***concatenation function****: given* $m, n \in \mathbb{N}$,

$$\langle m, n \rangle := \bar{m} B(n) \tag{B.5}$$

We call $\langle m, n \rangle$ *a* ***self-delimiting concatenation*** *of* m, n.

The usage of the self-delimiting concatenation is in a universal Turing machine or a universal finite codebook computer (UFCC), where the input is two natural numbers. We have to use a self-delimiting concatenation so that the string mn is uniquely identified as (m, n).

Definition 21 *A* ***Turing machine (TM)*** *consists of a finite program, called the* ***finite control****, capable of manipulating a linear list of cells, called the* ***tape****, using one access pointer, called the head. We refer to the two directions on the tape as right and left. The finite control can be in any one of a finite set of states Q, and each tape cell can contain a* 0*, a* 1*, or a blank* B*. The TM can perform the following basic operations:*

1. *it can write an element from* $A = \{0, 1, B\}$ *in the cell it scans; and*
2. *it can shift the head one cell left or right.*

After each step, the finite control takes on a state from Q*, and then decides an action according to a global list of* ***rules****. The rules have format* (p, s, a, q)*:* p *is the current state of the finite control;* s *is the symbol under scan;* a *is the next operation to be executed of type 1 or 2 designated in the obvious sense by an element from* $S = \{0, 1, B, L, R\}$*; and* q *is the state of the finite control to be entered at the end of this step. If the TM enters a state that does not appear as an entry* p *in the list of rules, then the TM* ***halts****. In particular, we define an extra state "reject" such that TM halts on "reject". We say that a TM* ***rejects*** x *if* T *inputs* x *and halts on "reject".*

Definition 22 (Formal version of Definition 6) *A Turing machine* T *is said to* ***compute*** *a function* $f : A \subset \mathcal{X} \to \mathcal{B}^*$*, if* T *rejects any input outside* A *and for all* $x \in A$*,* T *halts in a finite number of steps with* $f(x)$ *written on the tape.*

Intuitively, a TM T computes a function f if they have the same domain A, and for all $x \in A$, $T(x) = f(x)$.

Definition 23 *We define the* ***equivalence relation*** $\sim$ *between two Turing machines:* $S \sim T$ *if* S *and* T *halt on the same inputs (i.e. considered as functions, they have the same domain* L*) and for all* $x \in L$, $S(x) = T(x)$*. Namely,* S *and* T *compute the same function.*

Definition 24 (informal) *(Li et al., 2019) A universal Turing machine* U *is a Turing machine that can imitate the behavior of any other Turing machine* T*.*

In this paper, we mean "simulate" by "imitating the behavior of another Turing machine". We have not found any formal definition of "simulation". For the formalization of how a universal Turing machine simulates any other Turing machines, see Hennie and Stearns (1966).

B.2. Discrete probability theory

Definition 25 *(Li et al., 2019) Let* $\mathcal{B}$ *be a finite or countably infinite set of symbols. In this paper, we use* $\mathcal{B} = \{0, 1\}$*. A* ***cylinder*** *is a set* $\Gamma_x \subseteq \mathcal{B}^\infty$ *defined by*

$$\Gamma_x = \{x\omega : \omega \in \mathcal{B}^\infty\} \tag{B.6}$$

with $x \in \mathcal{B}^*$*. Let* $\mathcal{G} = \{\Gamma_x : x \in \mathcal{B}^*\}$ *be the set of all cylinders in* $\mathcal{B}^\infty$*. A function* $\mu : \mathcal{G} \to \mathbb{R}$ *defines a probability measure if*

$$\mu(\Gamma_\epsilon) = 1,$$
$$\mu(\Gamma_x) = \sum_{b \in \mathcal{B}} \mu(\Gamma_{xb}).$$

Consider the function $\mu' : \mathcal{B}^* \to \mathbb{R}$ defined by $\mu'(x) = \mu(\Gamma_x)$. Trivially from μ' we can reconstruct μ and vice versa. From now on we identify μ' with μ. Formally, we use the definition of measure below. One should keep in mind that our notation is shorthand for the original measure.

Definition 26 *A* ***finite cylinder of depth*** m *is defined by*

$$\Gamma_x^m = \{x\omega : \omega \in \mathcal{B}^m\}. \tag{B.7}$$

The probability space $\mathcal{X}$ we consider in Definition 5 can be redefined as $\mathcal{X} := \{\Gamma_x : x \in \mathcal{B}^*\}$.

Definition 27 *(Li et al., 2019) A function* $\mu : \mathcal{B}^* \to \mathbb{R}$ *is a* ***probability measure*** *(measure for short) if*

$$\mu(\epsilon) = 1 \quad \text{and} \quad \mu(x) = \sum_{b \in \mathcal{B}} \mu(xb), \tag{B.8}$$

for all $x \in \mathcal{B}^*$*. A* ***semi-measure*** *is a defective measure. A function* $\mu : \mathcal{B}^* \to \mathbb{R}$ *is a semi-measure if for all* $x \in \mathcal{B}^*$,

$$\mu(\epsilon) \leq 1,$$
$$\mu(x) \geq \sum_{b \in \mathcal{B}} \mu(xb).$$

Lemma 28 *When the precision of Z is fixed, the conditional independence set*

$$\{X \perp\!\!\!\perp Y|Z; \quad X, Y, Z \text{ are random variables in } \mathbb{P}\}$$

is decreasing as the precisions of X and Y increase.

Proof We will prove that if $X_{\_} \perp\!\!\!\perp Y|Z$, then $X \perp\!\!\!\perp Y|Z$.

For all $i \in \mathcal{B}$, for all (x, y, z) in the support of (X, Y, Z), $\mathbb{P}(xi, y, z)\mathbb{P}(z) = \mathbb{P}(xi, z)\mathbb{P}(y, z)$. Therefore,

$$\begin{aligned}
\mathbb{P}(x, y, z)\mathbb{P}(z) &= [\mathbb{P}(x0, y, z) + \mathbb{P}(x1, y, z)]\,\mathbb{P}(z) && \text{(B.9)}\\
&= \mathbb{P}(x0, z)\mathbb{P}(y, z) + \mathbb{P}(x1, z)\mathbb{P}(y, z) && \text{(B.10)}\\
&= \mathbb{P}(x, z)\mathbb{P}(y, z) && \text{(B.11)}
\end{aligned}$$

which implies $\mathbb{P}(X, Y|Z) = \mathbb{P}(X|Z)\mathbb{P}(Y|Z)$. ■

B.3. Compression in information theory

Here we recall some basic results in information theory. We follow Cover (1999).

Definition 29 *A **codebook**[13] (or source code) c for a random variable X is a mapping $\mathcal{X} \to \mathcal{B}^* = \{0, 1\}^*$. Let $c(x)$ denote the codeword corresponding to x and let $l(c(x))$ denote the length of $c(x)$.*

The expected length $L(c)$ of a codebook c for a random variable X with probability mass function $p(x)$ is given by

$$L(c) = \mathbb{E}[l(c(x))] = \sum_{x \in \mathcal{X}} p(x)l(x).$$

Example 2 *Let X be a random variable with the following distribution and codeword assignment:*

$$\begin{aligned}
\mathbb{P}(X = 1) &= \frac{1}{2}, \quad \textit{codeword } c(1) = 0\\
\mathbb{P}(X = 2) &= \frac{1}{4}, \quad \textit{codeword } c(2) = 10\\
\mathbb{P}(X = 3) &= \frac{1}{8}, \quad \textit{codeword } c(3) = 110\\
\mathbb{P}(X = 4) &= \frac{1}{8}, \quad \textit{codeword } c(4) = 111.
\end{aligned}$$

The entropy $H(X)$ of X is 1.75 bits, and the expected length $L(c) = E[l(X)]$ of this code is also 1.75 bits. Here we have a code that has the same average length as the entropy. We note that any sequence of bits can be uniquely decoded into a sequence of symbols of X. For example, the bit string 0110111100110 is decoded as 134213.

[13] In most literature in information theory (Shannon, 1948; Cover, 1999) the word "codebook" only denotes a code for $\mathcal{X}^N$ with N samples in the noisy-channel coding setting. We abuse the usage of this word in the source coding setting because we would like to stress that the source code itself also needs to be encoded and it also has a coding length, i.e. the coding length of the code(book).

Definition 30 *The **extension** c^* of a codebook c is the mapping from finite-length strings of $\mathcal{X}$ to finite-length strings of $\mathcal{B}^*$, defined by*

$$c^*(x_1x_2\cdots x_n) = c(x_1)c(x_2)\cdots c(x_n),$$

where $c(x_1)c(x_2)\cdots c(x_n)$ indicates concatenation of the corresponding codewords.

Example 3 *If $c(x_1) = 00$ and $c(x_2) = 11$, then $c(x_1x_2) = 0011$.*

Definition 31 *A code c is called **uniquely decodable** if its extension c^* is non-singular, namely, for all $x, x' \in \mathcal{X}^*$ sequences of letters in $\mathcal{X}$ such that $x \neq x'$, $c^*(x) \neq c^*(x')$.*

In other words, any encoded string in a uniquely decodable code has only one possible source string producing it.

Definition 32 *A codebook is called a **prefix code** or an instantaneous code if no codeword is a prefix of any other codeword.*

An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable. Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it.

We cannot assign short codewords to all source symbols and still be prefix-free. The set of codeword lengths possible for instantaneous codes is limited by the following inequality.

Lemma 33 (Kraft inequality) *For any instantaneous code (prefix code) over an alphabet of size D $(= 2$ in our case), the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality*

$$\sum_i D^{-l_i} \leq 1.$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

Corollary 34 *(Grünwald, 2007, p. 96) Let $\mathcal{X}$ be a finite or countable sample space. Let $\mathbb{P}$ be a probability distribution over $\mathcal{X}^n$, the set of sequences of length n. Then there exists a prefix code c for $\mathcal{X}^n$ such that for all $x^n \in \mathcal{X}^n, L_c(x^n) = -\log \mathbb{P}(x^n)$. c is called the code corresponding to $\mathbb{P}$.*

Similarly, let c' be a prefix code for $\mathcal{X}^n$. Then there exists a defective probability distribution $\mathbb{P}'$ such that for all $x^n \in \mathcal{X}^n$, $-\log \mathbb{P}'(x^n) = L'_c(x^n)$. $\mathbb{P}'$ is called the probability distribution corresponding to c'.

Theorem 35 (Shannon's source coding theorem) *(Cover, 1999, Thm. 5.4.1) Let $l_1^*, l_2^*, \ldots, l_m^*$ be optimal codeword lengths for a source distribution p and a D-ary alphabet, and let L^* be the associated expected length of an optimal code $(L^* = \sum p_i l_i^*)$. Then*

$$H_D(X) \leq L^* < H_D(X) + 1.$$

Theorem 35 corresponds to the following communication game: Alice and Bob share the same codebook $c : \mathcal{X} \to \mathcal{B}^*$, and Alice wants to transmit losslessly a sequence $x_1 \dots x_n$ iid sampled from $\mathbb{P}_X$ to Bob. She encodes the sequence into $c^*(x_1 \dots x_n) = c(x_1) \dots c(x_n)$ and sends the code sequence to Bob. Bob decodes the sequence using the same codebook. Theorem 35 proves that in this case, asymptotically the minimum expected binary coding length for a word $x \in \mathcal{X}$ sampled from $\mathbb{P}_X$ is around $H_2(X)$.

Suppose Alice and Bob share nothing except a programming language (C, python), or a universal Turing machine. In that case, the overall bits Alice needs to send can be two parts: a codebook, and a binary codeword sequence.

Definition 36 (Huffman coding program (MacKay, 2003)) *A Huffman coding program is a Turing machine that proceeds as follows:*

Input: the probability space $\mathcal{X}^d$, and a distribution $\mathbb{P}_X$ as a discrete function.

1. *Take the two least probable symbols in the alphabet $\mathcal{X}^d$. These two symbols will be given the longest codewords, which will have equal length and differ only in the last digit.*

2. *Combine these two symbols into a single symbol, and repeat.*

Output: a tree-structured codebook. We call the output of this program a ***Huffman code****.*

Theorem 37 (Huffman coding is optimal (Cover, 1999)) *If c^* is a Huffman code for $\mathbb{P}_X$ and c' is any other uniquely decodable code, $\mathbb{E}_{\mathbb{P}_X}[l(c^*(X))] \leq \mathbb{E}_{\mathbb{P}_X}[l(c'(X))]$.*

Therefore in the paper, we use the Huffman coding program to transform a multi-env distribution into a codebook. No matter which coding program we choose (e.g. Shannon-Fano-Elias code or arithmetic code), the expected codeword length for each $x \in \mathcal{X}^d$ is close to $\log \mathbb{P}_X(x)$. Since we can fix a coding program in a UFCC and still compute all finite codebooks, different choices of coding programs do not change significantly the theoretical result of model selection.

Appendix C. Proof in § 2

Lemma 2 ***(Identifiability implies uniqueness of solution of minimum cross-entropy)***[14]

Given a CBN model class $\mathcal{M}$, if $\mathcal{M}$ is identifiable and its distribution class is parametric, then

(1) For the observational CBN model, the solution of maximum likelihood $\arg\max_{M \in \mathcal{M}} \lim_{n\to\infty} \frac{1}{n} \log L(M|\mathcal{D}_n)$ *is unique. Equivalently, the minimizer of cross-entropy* $\arg\min_{(G,\theta)\in\mathcal{M}} \mathbb{E}_{\mathbb{P}^*_\theta}[-\log \mathbb{P}_\theta(X)]$ *is unique.*

(2) For the multi-env CBN model with uniform prior over environments, the solution of maximum likelihood $\arg\max_{M\in\mathcal{M}} \lim_{n\to\infty} \sum_{i=1}^{K} \frac{1}{n} \log L(M|\mathcal{D}^i_n)$ *is unique. Equivalently, there is a unique minimizer of the sum of cross-entropies across multi-env given by*

$$\arg\min_{\left((G_i)_{i\in[I]},(\theta_i)_{i\in[I]}\right)\in\mathcal{M}} \sum_{i=1}^{I} \mathbb{E}_{\mathbb{P}_{\theta^*_i}}[-\log \mathbb{P}_{\theta_i}(X)]. \tag{2.1}$$

[14] For readability we stay in the unconfounded setting and the strong version of identifiability. We can readily generalize Definition 1 and this lemma to identifiability up to an equivalence class, or generalize to the setting of causal representation learning, by changing the observed distribution to be a marginal distribution $\mathbb{P}_X$ of the model distribution $\mathbb{P}_{XZ}$, while forcing the invariance of $\mathbb{P}_{X|Z}$.

Proof (i) First, consider the observational CBN model class. Denote θ^* as the parameter of the distribution of which the marginal distribution generates $\mathcal{D}_n$, it is a solution of

$$\underset{(G,\theta)\in\mathcal{M}}{\arg\max} \lim_{n\to\infty} \frac{1}{n} \log L(\theta|\mathcal{D}_n). \tag{C.1}$$

For any $\theta \neq \theta^*$, by Definition 1, $\mathbb{P}_\theta$ and $\mathbb{P}^*_\theta$ are different on a μ-non-negligible set. So the random variable $\frac{p_\theta(X)}{p_{\theta^*}(X)}$ is non-degenerate. In addition, the log function is strictly convex, so by Jensen's inequality,

$$\mathbb{E}_{\mathbb{P}_{\theta^*}}\left[\log \frac{p_\theta(X)}{p_{\theta^*}(X)}\right] < \log \mathbb{E}_{\mathbb{P}_{\theta^*}}\left[\frac{p_\theta(X)}{p_{\theta^*}(X)}\right]. \tag{C.2}$$

The right-hand side is zero because

$$\mathbb{E}_{\mathbb{P}_{\theta^*}}\left[\frac{p_\theta(X)}{p_{\theta^*}(X)}\right] = \int_{\mathcal{X}} \frac{p_\theta(x)}{p_{\theta^*}(x)} p_{\theta^*}(x)dx = 1. \tag{C.3}$$

Therefore

$$\mathbb{E}_{\mathbb{P}_{\theta^*}}\left[\log p_\theta(X)\right] < \mathbb{E}_{\mathbb{P}_{\theta^*}}\left[\log p_{\theta^*}(X)\right]. \tag{C.4}$$

Equivalently, by applying the law of large numbers on eq. (C.4),

$$\lim_{n\to\infty} \frac{1}{n} \log L(\theta|\mathcal{D}_n) < \lim_{n\to\infty} \frac{1}{n} \log L(\theta^*|\mathcal{D}_n). \tag{C.5}$$

Therefore, maximum likelihood or minimum cross-entropy has a unique solution θ^*. Since we assume that $\mathcal{M}$ is identifiable, which means that each θ only has one underlying graph G, we conclude that there is only one solution (G^*, θ^*).

(ii) The proof for the multi-env CBN model class is the same, but with different sample space: $\mathcal{X}^{dI}$. It is a I-fold copy space for $\mathcal{X}^d$, with distributions from different environments lying in different dimensions. X_i^j is the i-th dimension in the original sample space $\mathcal{X}^d$ in the j-th environment. We use $\overline{\theta} = (\theta_1, \dots \theta_I)$ to denote the parameter for a distribution in $\mathcal{X}^{dI}$, which uniquely corresponds to the multi-env distributions $(\mathbb{P}_{\theta_i})_{i\in[I]}$. We use $\overline{\Theta}$ to denote the parametric family of all $\overline{\theta}$ that is generated from the model class $\mathcal{M}$. By assumption of interventional environments, in $\mathcal{M}$, $\mathbb{P}^i \perp\!\!\!\perp \mathbb{P}^j$ for all $i \neq j \in [I]$, any probability distribution in the parametric family $\overline{\Theta}$ can be written as follows:

$$\mathbb{P}_{\overline{\theta}}(X) = \prod_{i=1}^{I} \mathbb{P}_{\theta_i}(X^i) \tag{C.6}$$

Notice that on the left, X is a random variable in $\mathcal{X}^{dI}$.

The sum of cross-entropy

$$\sum_{i=1}^{I} \mathbb{E}[-\log \mathbb{P}_{\theta_i}(X)] = \mathbb{E}[-\log \mathbb{P}_{\overline{\theta}}(X)] \tag{C.7}$$

which is exactly the cross-entropy of the I-fold variable in $\mathcal{X}^{dI}$. Same as the proof (i), if there exists $i \in [I]$ such that $\theta_i \neq \theta_i^*$, then $\mathbb{P}_{\theta_i}$ and $\mathbb{P}^*_{\theta_i}$ are different on a μ-non-negligible set. Using the same argument of Jensen's inequality over $\overline{\theta}$, we infer that $\mathbb{E}_{\mathbb{P}_{\overline{\theta}^*}}\left[\log p_{\overline{\theta}}(X)\right] < \mathbb{E}_{\mathbb{P}_{\overline{\theta}^*}}\left[\log p_{\overline{\theta}^*}(X)\right]$. By eq. (C.7), this is equivalent to

$$\sum_{i=1}^{I} \mathbb{E}[-\log \mathbb{P}_{\theta_i^*}(X)] < \sum_{i=1}^{I} \mathbb{E}[-\log \mathbb{P}_{\theta_i}(X)] \tag{C.8}$$

Therefore, maximum likelihood or minimum cross-entropy has a unique solution $(\theta_i^*)_{i\in[I]}$. Since we assume that $\mathcal{M}$ is identifiable, which means that each tuple $(\theta_i^*)_{i\in[I]}$ only has one underlying tuple of graphs $(G_i)_{i\in[I]}$, we conclude that there is only one solution $((G_i)_{i\in[I]}, (\theta_i^*)_{i\in[I]})$.

■

Appendix D. Supplemental theory in § 3 Algorithmic causality

One example that reflects our motivation is the following: LLMs show impressive performance on the questions that require the use of composable mechanisms such as physical laws, reasoning, and mathematics, but their objective in pre-training is only minimizing cross-entropy of next word given the context (Gupta et al., 2023). By minimizing cross-entropy, can a learned model store and call some reusable mechanisms (conditional probability to certain precisions) just like copying the laws in different environments? Does compression mean that LLMs compress many similar scenarios into sparse mechanisms? To illustrate this point, consider two models A, B which are indistinguishable in inputs and outputs, both compute a function $F : (i, x) \mapsto f(x)$ for all i. On input $(1, x)$, A uses a neural network for all x and outputs $f(x)$, and on input $(2, x)$, A copies the neural network f as COPY$(f)(x)$, where COPY(f) can be stored as an address in memory pointing to the neural network f. Here, 1 or 2 denote the context or environment. B utilizes a sophisticated neural network to compute F. Since A and B are undistinguishable as functions, their cross-entropies are also undistinguishable (recall that entropy is the minimal length of encoding iid data *given* a model). Consequently, minimizing cross-entropy does not ensure the presence of any sparse structure within the model itself. We have to go beyond entropy and identifiability, by taking model length into account.

The idea of "copying mechanisms" is the motivation behind § 3 (step 2 and 3 in Definition 9) and Proposition 46. Instead of treating all the parameters equally, we can decompose a model into some computation steps. First, the model creates some "raw mechanisms"(Definition 7) without assigning them to a variable in $\mathcal{X}^d$, for example "something falls at the acceleration $9.8m/s^2$"; second, the model apply the raw mechanisms to some concrete objects or values: "An apple in Europe falls at the acceleration $9.8m/s^2$", "An iron ball in Asia falls…". The raw mechanism "something falls at the acceleration $9.8m/s^2$" is copied multiple times and "something" is replaced by objects in different places or contexts by a feature mechanism (Definition 8), such as "apple/iron balls in Europe/Asia $\mapsto$ something" (Example 1 Step 2). The gravitational acceleration in Europe and Asia are slightly different, but compression will decide whether saving one or two different mechanisms in the model is better, by balancing the model coding length and data-to-model coding length. The model that compresses the Internet optimally would save the variables in the following way: "**if** something is on earth, **then** it falls at the acceleration of $9.8m/s^2$". Algorithmic causal statements are such reusable "if...then..." judgments that emerge from compression.

It is important to notice that there is no interventional data or identifiability here: compression prefers to put "on earth" as a cause of "acceleration value", not because we did single-node interventions on those two variables respectively. Available training data for LLMs does not include intervention targets or environment variables. The data is multi-env, in the sense that contexts are always different in each data sentence (country, temperature, etc.), but there is only statistical dependence (correlation) in the data. Since there are no intervention targets or hard constraints over model class, no identifiability in causal literature is applicable. However, human and compression algorithms can still believe that it is “on earth” that “causes” the ”acceleration value”, not the other way around. The rest of the paper explains how compression can prefer one causal statement over others.

Example 4 (multi-env CBN) *Consider a multi-env system:* $\mathbb{P}(X_1, X_2, E = 1) = \mathbb{P}(X_2|X_1)\mathbb{P}(X_1|E = 1)\mathbb{P}(E = 1)$*,* $\mathbb{P}(X_1, X_2, E = 2) = \mathbb{P}(X_2|X_1)\mathbb{P}(X_1|E = 2)\mathbb{P}(E = 2)$*.*

Now we give an example of CFMP α *that computes this system. Step 1 in Def 9,* α *generates the following: Probabilistic mechanisms (Definition 7):* $f_1(\cdot|\cdot)$*,* $f_2(\cdot|\cdot)$*,* $f_3(\cdot)$*.*

Feature mechanisms (Definition 8): $\phi_1 : (x_1, x_2, e) \mapsto x_1$*,* $\phi_2 : (x_1, x_2, e) \mapsto x_2$*,* $\phi_3 : (x_1, x_2, e) \mapsto e$*.*

Step 2, α *featurizes the mechanisms:* $f_1 \mapsto f_1(\phi_2(\cdot)|\phi_1(\cdot))$*,* $f_2 \mapsto f_2(\phi_1(\cdot))$*,* $f_3 \mapsto f_3(\phi_3(\cdot))$*.*

Step 3, given any $z = (x_1, x_2, e)$*, compute the probability value*

$$\mathbb{P}(z) = f_1(\phi_2(z)|\phi_1(z))f_2(\phi_1(z))f_3(\phi_3(z)).$$

Example 5 (More examples in Example 1)

1. *A* ***context-specific Bayesian network*** *model (Boutilier et al., 1996) computing* $\mathbb{P}$ *is a CFMP* α *that has the same definition as CBN except (b) in step 3 of CBN: for different points* $x, x' \in \mathcal{X}^d$*,* α *is allowed to use different sets of featurized mechanisms depending on* x, x'*, see footnote 6.*

2. *A* ***transitive*** G***-equivariant learning*** *model computing* $\mathbb{P}$ *on* $\mathcal{X}^2$ *is a CFMP* α *such that*

   (a) *in* $\mathcal{P}_\alpha$ *there exists a probabilistic mechanism* $f : \mathcal{X} \to \mathcal{B}^n$ *modeling a function* $x \mapsto \mathbb{P}(\tilde{x}|g.x)$ *where* $\tilde{x}$ *is an arbitrary fixed point in* $\mathcal{X}^d$*. For each* $(x, x') \in \mathcal{X} \times \mathcal{X}$*, there exists* $g \in G$ *such that* $x = g.\tilde{x}$ *and* $\mathbb{P}(x|x') = f(x|x') = f(g^{-1}.x|g.x') = f(\tilde{x}|g.x') = \mathbb{P}(\tilde{x}|g.x')$*.*
   There exists $|G|$*-many feature mechanisms in* Φ_α *computing the group actions on* $\mathcal{X}$*: for all* $g \in G$*,* $\phi_g : x \mapsto g.x$*.*

   (b) *In step 2,* α *generates* $|G|$*-many featurized mechanisms using* f *and* $(\phi_{g^{-1}}, \phi_g)$ *for all* $g \in G$*:* $f \mapsto f(\phi_{g^{-1}}(\cdot)|\phi_g(\cdot))$*.* α *also generates a featurized mechanism* $h(\phi_g \circ \pi_1(\cdot))$ *such that* $\mathbb{P}_1(\phi_g(x_1)) = h(\phi_g \circ \pi_1(x))$

   (c) *In step 3,* α *only uses those featurized* $f(\phi_{g^{-1}}(\cdot)|\phi_g(\cdot))$ *for computing the value variable* $y \in \pi_2(\mathcal{X}^2)$*: for all* $x \in \mathcal{X}^2$*, there exists a featurized mechanism* h *and* $g \in G$ *such that* $\mathbb{P}(x) = \mathbb{P}_2(\phi_{g^{-1}}(x_2)|\phi_g(x_1))\mathbb{P}_1(\phi_g(x_1)) = f(\phi_{g^{-1}} \circ \pi_2(x)|\phi_g \circ \pi_1(x))h(\phi_g \circ \pi_1(x))$*.*

3. *A* G***-equivariant learning*** *model computing* $\mathbb{P}$ *on* $\mathcal{X}^2$ *is a CFMP* α *that has the same definition as the transitive* G*-equivariant learning model except that in (a):*

   (a) *in* $\mathcal{P}_\alpha$ *there exists a probabilistic mechanism* $f : \mathcal{X}/G \times \mathcal{X} \to \mathcal{B}^n$ *where each element in* $\mathcal{X}/G$ *is a fixed representative element of an equivalence class.*[15] *For each* $(x, x') \in$

[15] Different from the common usage of the quotient in algebra, here each representative in $\mathcal{X}/G$ is a given point in $\mathcal{X}$.

$\mathcal{X} \times \mathcal{X}$, there exists $g \in G$ and $\tilde{x} \in \mathcal{X}/G$ such that $x = g.\tilde{x}$ and $\mathbb{P}(x|x') = f(x|x') = f(g^{-1}.x|g.x') = f(\tilde{x}|g.x') = \mathbb{P}(\tilde{x}|g.x')$.

Definition 38 (Algorithmic causality) *Given a class of CFMPs, we say that according to a model selection method, $\phi(X)$ **algorithmically causes** $\psi(X)$ **locally at** $x \in \mathcal{X}^d$ if the method selects a CFMP α such that, in the third step in α the featurized mechanisms for x include one mechanism $f(\psi(X)|\phi(X))$ in which $\psi(X)$ is a value variable and $\phi(X)$ is a conditional variable (Definition 7), and do not include mechanism with the opposite direction. Similarly, we say that $\phi(X)$ **algorithmically causes** $\psi(X)$ if the above assumption holds for all $x \in \mathcal{X}^d$.*

Moreover, if the class of CFMP allows hidden-variable mechanisms (Definition 9, 2.) in certain CFMPs and if the method selects a CFMP α such that in the third step in α the featurized mechanisms for $x \in \mathcal{X}^d$ include one hidden-variable mechanism $(x, x') \mapsto f(\phi(x)|\psi(x'))$ and do not include the mechanisms with the opposite direciton, then we say that the hidden variable $\psi(X')$ algorithmically causes $\phi(X)$ locally at $x \in \mathcal{X}^d$, and $\psi(X')$ algorithmically causes $\phi(X)$ if it holds for all $x \in \mathcal{X}^d$.

For example, in causal representation learning we are typically interested in the latent causal graph, which is in our language the causal relationships between $(\pi_i \circ \psi(X'))_{i\in[k]}$, where k denotes the dimension of the latent space. Our definition of causality is flexible enough to model the causal relationships between *imagined* variables, i.e. between $\phi(X)$ and $\psi(X)$ with ϕ, ψ being any feature mechanisms. Different from the measure-theoretic foundation of interventional causality (Park et al., 2023), we do not need to predetermine the causal variables and their spaces. Our choice of defining causal variables can emerge from the model selection through the choice of feature mechanisms. In fact, the dimension d and precision m in $\mathcal{X}$ are only used to illustrate the abstract concepts of Definition 9; equivalently, one can replace $\mathcal{X}^d$ by a binary input tape of arbitrary fixed length, since in a computer or UFCC there are only processes of binary variables at the basic level.

Appendix E. Supplemental theory in § 4 Learning algorithmic causality by compression

E.1. Kolmogorov complexity

We review Kolmogorov complexity, which led to the idea of two-part code (Li et al., 2019). This inspired our idea of finite codebook complexity. We leave some definitions in computation theory in § B.1.

Definition 39 *(Kolmogorov, 1968; Li et al., 2019) (First version) For any $x \in \mathbb{N}$, the **Kolmogorov complexity** of x w.r.t. the universal Turing machine U is defined as*

$$C_U(x) = \min_{T \in \{\textit{Turing machines}\}} \{l_U(T)|U(T) = x\} \tag{E.1}$$

where l_U is a mapping from the class of all Turing machines to $\mathbb{N}$ such that for each $n \in \mathbb{N}$ there are less than 2^n Turing machines T such that $l_U(T) \leq n$.

*(Second version) equivalently, the **Kolmogorov complexity** of x w.r.t. the universal Turing machine U can also be defined as*

$$C_U(x) = \min_{n\in\mathbb{N}} \{l_U(n)|U(n) = x\} \tag{E.2}$$

where l_U is a monotonically increasing map from $\mathbb{N} \to \mathbb{N}$ such that for any $n \in \mathbb{N}$, $l_U(2^n) \leq n$.

In the second version of the definition, the input of l_U is not a Turing machine, but an index of a Turing machine. It is important to note that each universal Turing machine (UTM) defines a computable bijective mapping $\mathbb{N} \to \{\text{Turing machines}\}$: $U(0) = T_0, U(1) = T_1, \ldots,$[16] which is called an **effective enumeration of TMs**. A UTM does not have to take the literal description of a Turing machine as input. There exists a UTM U for which a TM T with 5 states has the length $l_U(T) = 1$. In Li et al. (2019) and much of the literature, people use l instead of l_U, which can be somewhat confusing because they implicitly assume that readers are aware that each UTM defines a different effective enumeration over all TMs. In this paper, we use l to denote the **literal length function** that maps a natural number n (or its equivalent binary string $B(n)$, see Definition 19) to $\lfloor \log_2(n+1) \rfloor$, and we use l_U in both versions, which can be distinguished automatically by their input.

Lemma 40 *(Kolmogorov, 1968; Li et al., 2019) There is an additively optimal universal Turing machine U, i.e. for all UTM V and all x, $C_U(x) \leq C_V(x) + O(1)$.*

Proof We will construct a UTM U. It needs inputs in the form

$$\langle n, p \rangle = \underbrace{11\ldots1}_{l(n) \text{ times}} \quad 0\, B(n) B(p)$$

where $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathcal{B}^*$ **is the self-delimiting concatenation (Definition 20)**[17], and $B(\cdot)$ is the **binary lexicographic code** defined in Definition 19. U lists all TMs $T_0, T_1, T_2, \ldots$. Define $U(\langle n, p \rangle) := T_n(p)$. Namely, the UTM U first receives the self-delimiting code for n, then it simulates the Turing machine T_n, on the input number p.

For any UTM V, there exists n such that $V = T_n$. Suppose $p^* \in \arg\min_{p \in \mathbb{N}} \{l(p) : T_n(p) = x\}$, then $C_U(x) \leq l(\langle n, p^* \rangle) = 2l(n) + 1 + l(p^*) = 2l(n) + 1 + C_{T_n}(x) = c_{U,V} + C_V(x)$ where $c_{U,V} := 2l(n) + 1$ only depends on U and V.

■

This Lemma implies that for all such additively optimal TM, we can choose any of them with a coding length different by $O(1)$.

Definition 41 *(Li et al., 2019, Def. 2.1.1) Given an additively optimal UTM U, given $x, y \in \mathbb{N}$, the conditional Kolmogorov complexity $C_U(x|y)$ is defined as*

$$C_U(x|y) = \min_{p \in \mathbb{N}} \{l(p) : U(\langle y, p \rangle) = x\} \tag{E.3}$$

Lemma 42 (Kolmogorov complexity can be written as two-part code (Li et al., 2019)) *Suppose U is an additively optimal UTM taking inputs $\langle n, p \rangle$ as in the proof of Lemma 40, then*

$$\begin{aligned} C_U(x) &= \min\{2l_U(T) + l_U(p) : T(p) = x\} && \text{(E.4)} \\ &= \min\{2l_U(T) + C_U(x|T) : T \in \{T_0, T_1, \ldots\}\} && \text{(E.5)} \end{aligned}$$

*where $2l_U(T)(+1)$ is the binary bit length of the **self-delimiting code** (Definition 20) of T in the effective enumeration of U.*

[16] By convention, $U(0)$ is often defined as U itself (Li et al., 2019).

[17] Its use is simply to ensure that the concatenated numbers (np) are uniquely identified.

Proof (Adapted from (Li et al., 2019, 2.1.1)) By definition, $C_U(x) = \min\{l(q) : U(q) = x\}$, with $q = \langle n, p\rangle$ as in Lemma 40. If q^* is a number such that $l(q^*) = C_U(x)$, then there exists n^*, p^* s.t. $C_U(x) = l(q^*) = 2l(n^*) + 1 + l(p^*) = 2l_U(T_{n^*}) + 1 + l(p^*)$ and $T_{n^*}(p^*) = x$, where $l_U(T_{n^*}) = l(n^*)$ is the length of the index of the Turing machine T_{n^*} in U and its self-delimiting coding length is $2l(n^*) + 1$ according to Definition 20. ■

Intuitively, the $l_U(T)$ part of the code squeezes out the regularities in x. $C_U(x|T)$ is irregularities, or random aspects, of x relative to that Turing machine. Since most strings are algorithmically random or incompressible (Li et al., 2019, Thm. 2.2.2), minimum cross-entropy is often the shortest coding length for a sequence x *given* the distribution computed by T that achieves the minimum cross-entropy (Li et al., 2019, Thm. 8.1.2). This idea has led to the two-part code objective in statistical inference (Grünwald, 2007), where the irrgularity part is upper bounded by the Shannon code of an iid sequence $c(x_1 \ldots x_n) := \sum_{i=1}^{n} -\log \mathbb{P}(x_i)$ *given a codebook*, but how to bound the regularity part, $l_U(T)$, to our knowledge, has not been discussed in previous literature.

E.2. Examples of UFCC and non-UFCC

Example 6 *Some examples of UFCC:*

- *There exists a Turing machine $V_{\mathcal{C}}$ that takes any element in a finite class $\mathcal{C}$ of CFMP as input and combines it with a Huffman coding program to compute a Huffman code, then decodes an integer p which is equivalent to an input binary sequence $B(p)$.*
- *There exists a Turing machine V_1 that takes a (multi-env) discrete distribution table as input and combines it with a Huffman coding program to compute a Huffman code, then decodes p.*
- *There exists a Turing machine V_2 that inputs any FCM as a table and then decodes p using that FCM.*

Example 7 *Note that the following Turing machine is not a UFCC:*
Given $\mathcal{X}^d$ of precision m, V_{halt} inputs two integers:

- *an index k of any Turing machine that halts at any point in $\mathcal{X}^d$ and computes a finite codebook;*
- *an integer p which is equivalent to a binary sequence as codeword.*

The reason is a direct reduction from the halting problem (§ E.2):

Lemma 43 *Given $l \in \mathbb{N}$, the set of Turing machines that halt at any input of length l and compute a finite codebook is undecidable.*

Proof We call the set above $\mathcal{F}$, and suppose $\mathcal{F}$ is decidable. Then there exists a Turing machine D which inputs the index k of any Turing machine T_k, and outputs a decision whether $T_k \in \mathcal{F}$.

We reduce the halting problem to the above decision problem. Given any Turing machine M and input x, we construct a new Turing machine $T_{M,x}$:

```
On input 0, ignore the input 0 and simulate $M$ on $x$
If $M$ halts on $x$, then delete the output of $M$ and output 0
If $M$ does not halt on $x$, then rejects
```

We run the TM D on $T_{M,x}$. By construction, $T_{M,x}$ either computes a trivial codebook $0 \mapsto 0$, or rejects. If D accepts $T_{M,x}$, then $T_{M,x}$ computes $0 \mapsto 0$ and halts. If D rejects $T_{M,x}$, then M does not halt on x.

Therefore D decides the halting problem, which is a contradiction. ■

Therefore, it is necessary to constrain the class of FCMs when defining UFCC, while not constraining the class of finite codebooks or probability distributions that FCMs can compute.

E.3. Comparisons among UFCCs

Are some UFCCs better than others? We are not interested in finding the UFCC that achieves the smallest FC complexity for all data, because it is often not computable. Instead, we are interested in finding some UFCCs that are good at model selections, i.e. such a UFCC should not consider all FCMs to be equally preferable to select.

For the Kolmogorov complexity, the choice of UTM is not important because all the additively optimal UTMs are equivalent up to $O(1)$ (Lemma 40). For UFCC it is not the case.

Consider the following extreme example: a UFCC U_{unif} first takes m, d, n as input where m denotes the precision of $\mathcal{X}$, i.e. $|\mathcal{X}| = 2^m$, and d denotes the number of variables, and n denotes the precision of discrete distribution values. Let U_{unif} encode any distribution by a table, where each row is a distribution value for a point in $\mathcal{X}^d$. Suppose the rows are well ordered so we do not need to encode the points for simplicity. Then U_{unif} only needs to encode $(2^m)^d$ numbers, with each number occupying n bits. After coding the distribution values, U_{unif} uses a Huffman coding program to turn it into a prefix code.[18] Therefore, for any codebook g, any binary codeword sequence $B(p)$ is decoded by FCMs with the same model length, i.e. same $l_{U_{\text{unif}}}(T)$ in eq. (4.4). For any (m, n, d), U_{unif} gives a uniform prior over all codebooks that are Shannon codes of a distribution on $\mathcal{X}^d$ with precision (m, n). Using U_{unif} as UFCC, the objective eq. (4.4) is equivalent to maximum likelihood. Namely, from the perspective of U_{unif}, no codebook is simpler or more preferable than another.

Back to our question: are some UFCCs better than others? We conjecture that a criterion for good UFCC is: for any additively optimal UTM, a good UFCC U should have a similar landscape (i.e. the order) as the UTM in the right figure of Figure 2. Namely, a good UFCC should preserve the order of codebooks in a certain UTM.

Appendix F. Supplemental results and proofs in § 5 Case studies

F.1. Lemmata

Lemma 44 (Bernoulli's Inequality (Carothers, 2000)) *If $a > -1$, $a \neq 0$, then $(1+a)^n > 1+na$ for any integer $n > 1$.*

Lemma 45 *Given $x \in \mathbb{Q} \cap [0, 1)$ in binary precision n, the minimum number of bits k required such that for all exact factorization $x = \prod_{i=1}^{l} y_i$ with $y_i \in \mathbb{Q} \cap [0, 1)$, $\prod_{i=1}^{l} y_i|_k$ has the same binary representation as x with truncated precision n, i.e.*

$$0 \leq x - \prod_{i=1}^{l} y_i|_k \leq 2^{-n-1},$$

[18] Notice that U_{unif} does not save the Huffman code on each point in $\mathcal{X}^d$, instead, it saves the distribution value on each point. The program that inputs a distribution and outputs a Huffman code is constant w.r.t. the distribution.

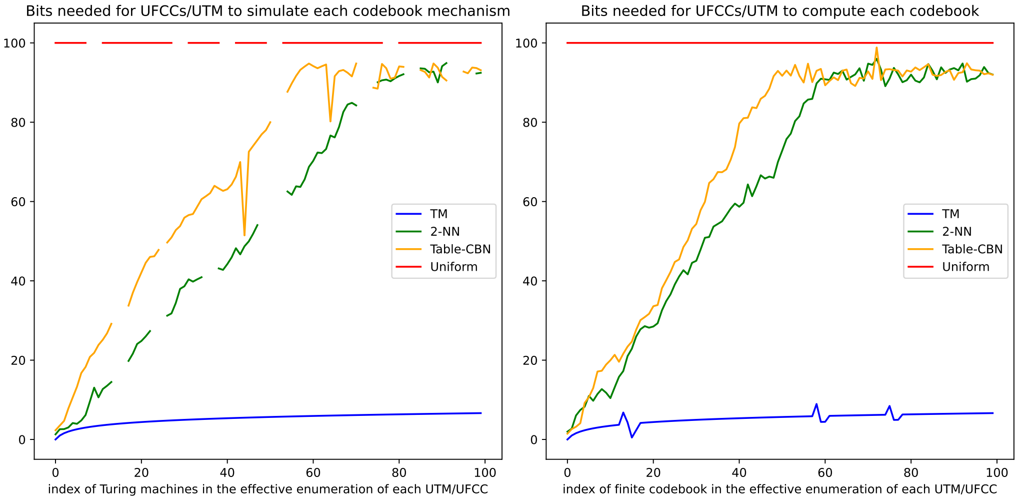

Figure 2: Given (m, n, d), we consider the codebooks on $\mathcal{X}^d := (\mathcal{B}^m)^d$ with precision (m, n). The curves are fictitious for illustration, since some of them are not computable. Left figure: The x-axis is the index of Turing machines in an effective enumeration of all Turing machines that can compute a codebook. The y-axis is the coding length using a universal Turing machine or a UFCC. Right figure: The x-axis is the index of codebooks in an effective enumeration of all codebooks. The y-axis is the minimum coding length of the Turing machine or FCM that the universal Turing machine or UFCC can simulate. Blue line: Turing machines simulated by an arbitrary universal Turing machine. Green line: FCMs simulated by a universal two-layer neural network computer. Yellow line: FCMs simulated by U_{TabCBN} defined in Proposition 16. Red line: FCMs simulated by U_{unif} defined above.

is at most $2n + \log_2 l + 3$.

Proof

Let $(x_i)_i$ be the approximations of y_i with k bits of precision, i.e. $(y_i|_k)_i$, so:

$$0 \leq y_i - x_i \leq \delta := 2^{-k} \tag{F.1}$$

Define the relative errors:

$$\epsilon_i = \frac{y_i - x_i}{y_i} \tag{F.2}$$

Case 1: $y_i \geq 2^{-n-2}$ for all i. Then we have:

$$\epsilon_i \leq \frac{\delta}{2^{-n-2}} = 2^{n-k+2} =: \epsilon \tag{F.3}$$

We aim to bound the error:

$$\prod_{i=1}^{l} y_i - \prod_{i=1}^{l} x_i = x\left(1 - \prod_{i=1}^{l}(1-\epsilon_i)\right) \leq 1 - \prod_{i=1}^{l}(1-\epsilon_i) \leq 1 - (1-\epsilon)^l \leq l\epsilon \tag{F.4}$$

where the rightmost inequality is by Lemma 44.

We want to find k such that the RHS is upper bounded by 2^{-n-1}:

$$l \cdot 2^{n-k+2} \leq 2^{-n-1} \tag{F.5}$$

which is equivalent to

$$k \geq 2n + \log l + 3. \tag{F.6}$$

Case 2: There exist $i \in [d]$ such that $y_i < 2^{-n-2}$ for all i. Then we have:

$$0 \leq \prod_{j=1}^{l} (y_j|_k) \leq y_i|_k < 2^{-n-2} \tag{F.7}$$

and

$$0 \leq x = \prod_{j=1}^{l} y_j \leq y_i|_k < 2^{-n-2}. \tag{F.8}$$

Therefore,

$$0 \leq x - \prod_{i=1}^{l} y_i|_k < 2^{-n-1} \tag{F.9}$$

■

F.2. Proofs in § 5

Proposition 16 *(Causal factorization is shorter than encoding the joint distribution) Suppose $\mathbb{P}$ is supported on $\mathcal{X}^d \times [I]$ with precision (m, n) and $\mathbb{P}(X, e_i) = \mathbb{P}^i(X_1|X_{S_i}, e_i)\mathbb{P}(X_2, \ldots, X_d)\mathbb{P}(e_i)$ (w.l.o.g. suppose $\mathbb{P}(e_i) = \frac{1}{I}$) and $(S_i)_{i\in[I]}$ are subsets of $[d]$ with $|S_i| < d-1$. Suppose $I = 2^{\log I} \leq 2^n$. Denote α, β as two CFMPs simulated by U_{TabCBN} and computing the same $\mathbb{P}$, while α is a CBN model (Example 1 (1)) proceeds by separately saving and calling factorized mechanisms, and β is a statistical density estimator (Example 1 (4)) which proceeds by encoding the whole multi-env distribution in $\mathcal{P}_\beta$. Then $l_{U_{TabCBN}}(\alpha)(m, d) = o(l_{U_{TabCBN}}(\beta)(m, d))$ as $m \to \infty$ or $d \to \infty$.*

Proof We construct a CFMP α, following Definition 9:

First step, α generates $\mathcal{P}_\alpha$, a list of probability tables:

- for all $i \in [d]$, f_i computes: $\mathbb{P}^i(\cdot|\cdot) : \mathcal{X} \times (\mathcal{X}^{|S_i|} \times [I])$, i.e. the shifted mechanisms of X_1;
- f_{d+1} computes the $X_2, \ldots, X_d$ marginal of $\mathbb{P}$;
- f_{d+2} : Unif$[I]$, a uniform prior over environments.

We want the probabilistic mechanisms output precise enough probability values such that the joint distribution computed at the end of step 3 is lossless. By Lemma 45, the output precision in each probabilistic mechanism needs $2n + \log 2 + 3 = 2n + 4$ bits.

For each $i \in [d]$, f_i is a table with input space at most $(\mathcal{B}^m)^{1+(d-2)} \times \mathcal{B}^{\log I}$, and output space $\mathcal{B}^{2n+4}$. The coding length of f_i as a table is at most $(2n + 4)(2^m)^{d-1} I$. The coding length of f_{d+1} is $(2n + 4)(2^m)^{d-1}$. The coding length of f_{d+2} is $(2n + 4)I$, since $\frac{1}{2^n} \leq \frac{1}{I} = \frac{1}{2^{\log I}}$ by assumption. In total we need $(2n + 4)((2^m)^{d-1} I + (2^m)^{d-1} + I)$ bits for $\mathcal{P}_\alpha$.

By assumption in UFCC, Φ_α only contains projections. Since the cardinal of the power set of the nodes is 2^{d+1}, we need $d+1$ bits for each feature mechanism. In total we need $(I+3)(d+1)$ bits for Φ_α.

Second step, we assign $I+3$ feature mechanisms to $I+2$ probabilistic mechanisms, each of which has two cases (conditional or value) for filling the feature mechanism. So the whole mapping costs $2(I+2)\log(2I+3)$ bits.

Third step, for each $(x,e) \in \mathcal{X}^d \times [I]$, α computes $\mathbb{P}(x,e) = f_e(x_1|x_{S_e})f_{d+1}(X_2,\ldots,X_d)f_{d+2}(e)$. We need $O(I\log I)$ bits because we only need to assign I-many mechanisms $(f_i)_{i\in[I]}$ to I-many environments, with other mechanisms being the same for all $(x,e) \in \mathcal{X}^d \times [I]$.

The Huffman coding mechanism Definition 15 that turns a distribution into a Huffman code is of constant coding length. In total, the coding length of α is $O(nI(2^m)^{d-1} + Id)$.

Define β as the CFMP that encodes the whole multi-env distribution: $\mathcal{P}_\beta$ has only one element: a probability table $\mathbb{P}$ of precision n. This costs $nI2^{m^d}$ bits. Φ_β only contains the identity, which costs $d+1$ bits, same as those elements in Φ_α. Since the joint distribution is already represented, all the following steps are trivial and only cost constant bits. Therefore, $l_{U_{\text{tabCBN}}}(\alpha)(m,d) = o(l_{U_{\text{tabCBN}}}(\beta)(m,d))$. ■

Proposition 18 *(Invariance factorization is shorter than Markov factorization) Suppose $\mathbb{P}$ is supported on $\mathcal{X}^2$ and $\mathbb{P}(X) = \mathbb{P}(X_1|\phi(X_2))\mathbb{P}(X_2)$, where $\phi : \mathcal{X} \to \mathcal{X}/G$ is the quotient map of a certain group action G that only acts on one dimension in $\mathcal{X}^2$ and leaves the other dimension unchanged. Suppose the number of orbits $\mathcal{X}/G$ is independent of the precision m. Denote α, β as two CFMPs simulated by U_{TabInv} and computing the same $\mathbb{P}$, while α is a G-invariant learning model (Example 1 (3)) which factorizes $\mathbb{P}$ into $\mathbb{P}(X) = f_1(\pi_1(X)|\phi\circ\pi_2(X))f_2(\pi_2(X))$, and β is a CBN model (Example 1 (1)) which factorizes $\mathbb{P}$ into $\mathbb{P}(X) = f_1'(\pi_1(X)|\pi_2(X))f_2(\pi_2(X))$ for certain f_1, f_1', f_2. Then $l_{U_{\text{TabInv}}}(\alpha)(m) = o(l_{U_{\text{TabInv}}}(\beta)(m))$.*

Proof The proof is trivial because the steps in α and β are the same only except for $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$.

$\mathcal{P}_\alpha$ consists of two tables: $f_1(\cdot|\cdot) : \mathcal{X} \times \phi(\mathcal{X}) \to \mathcal{B}^n$, $f_2 : \mathcal{X} \to \mathcal{B}^n$. By Lemma 45, each probability value in each probabilistic mechanism table needs $2n + \log 2 + 1 = 2n + 4$ bits. Therefore, f_1 needs $(2n+4)2^{m+c}$ bits, f_2 needs $(2n+4)2^m$ bits, where c is a constant independent of m.

$\mathcal{P}_\beta$ consists of two tables: $f_1'(\cdot|\cdot) : \mathcal{X} \times \mathcal{X} \to \mathcal{B}^n$, $f_2 : \mathcal{X} \to \mathcal{B}^n$. f_1 needs $(2n+4)2^{m^2}$ bits, f_2 needs $(2n+4)2^m$ bits.

Therefore, $l_{U_{\text{TabInv}}}(\alpha(m)) = o(l_{U_{\text{TabInv}}}(\beta(m)))$.

■

F.3. Formal version of Proposition 17

Consider a UFCC U_{CompCBN} which is defined as follows: for all α,

1. $\mathcal{P}_\alpha$ is compressible and finite, of cardinal M: there exists a Turing machine T such that for all $f \in \mathcal{P}_\alpha$, $C_{U_{\text{CompCBN}}}(f|T) = \log M \leq l_{U_{\text{TabCBN}}}(f)$. Namely, we do not need to store each probabilistic mechanism as a table. Instead, α can run the subprogram T in step 1, and then use a uniform code $\log M$ to encode each probabilistic mechanism. For example, we can write a Python program to generate a binomial distribution $B(n,p)$ with only two parameters, instead of storing each probabilistic map as a table.

2. Same as in U_{TabCBN}, Φ_α is restricted to projections on variables: for d variables, there are 2^d possible projections, encoded uniformly.
3. We consider two strategies for implementing step 3 of CFMPs in U_{CompCBN}. We assume that we want to use N featurized mechanisms which are elements of a subset of size k of all M available featurized mechanisms.
   (a) Strategy 1: Directly select N mechanisms from M mechanisms, i.e., encode the selected mechanisms by a map $[N] \to [M]$.
   (b) Strategy 2: First, write down k, the number of featurized mechanisms that are used. Then choose the k selected mechanisms encoded by a map $[k] \to [M]$. Moreover, encode the final assignment of the N mechanisms through a (surjective) map $[N] \to [k]$.

Under this UFCC, we prove that the factorization that uses the sparsest mechanism shifts compresses best among all factorizations:

Proposition 46 *In $U_{CompCBN}$, for $k = o(N)$ and $k < \frac{M}{2}$, strategy 2 has a shorter coding length than strategy 1, and the difference in coding length between strategy 1 and strategy 2 is decreasing in this range of k.*

Proof There are $\binom{M}{k}$ possibilities for choosing k mechanisms from $\mathcal{P}_\alpha$. There are $k!S_k^N$ possibilities for subjective functions $[N] \to [k]$, where S_k^N is the Stirling number of the second kind. Given k, the bits for step 2 in strategy 2 is $\log \binom{M}{k} + \log(k!) + \log k$. The bits for step 2 in strategy 1 are $N \log M$. So the difference between strategy 2 and strategy 1 is

$$A(k) = \log \binom{M}{k} + \log S_k^N + \log(k!) + \log k - N \log M \tag{F.10}$$

So the difference between $A(k+1)$ and $A(k)$ is

$$A(k+1) - A(k) = \log\left(\frac{M-k}{k+1}\right) + \log\left(\frac{S_{k+1}^N}{S_k^N}\right) + \log(k+1) + \log\left(\frac{k+1}{k}\right) \tag{F.11}$$

$$= \log\left((M-k) \cdot \frac{S_{k+1}^N}{S_k^N} \cdot \frac{k+1}{k}\right). \tag{F.12}$$

For $k = o(N)$, we have an approximation for Stirling number of the second kind: $\lim_{N\to\infty} S_k^N = \frac{k^N}{k!}$. Then

$$\lim_{N\to\infty} \frac{S_{k+1}^N}{S_k^N} = \frac{\frac{(k+1)^N}{(k+1)!}}{\frac{k^N}{k!}} = \frac{(k+1)^N}{k^N} \cdot \frac{k!}{(k+1)!} = \left(1 + \frac{1}{k}\right)^N \cdot \frac{1}{k+1} \tag{F.13}$$

In order that eq. (F.12) is greater than zero, by eq. (F.13) we need for any $k = o(N)$,

$$\frac{M-k}{k}\left(1 + \frac{1}{k}\right)^N > 1 \tag{F.14}$$

$$M > \frac{k}{(1+\frac{1}{k})^N} + k, \tag{F.15}$$

which holds when $k < \frac{M}{2}$.
Therefore, $A(k)$ monotonically increases when $k = o(N)$ and $k < \frac{M}{2}$. By eq. (F.10),

$$A(1) = (1 - N)\log M + \log 2 + 1 < 0 \tag{F.16}$$

when N is sufficiently large. The conclusion of the proposition is obtained from $-A$. ■

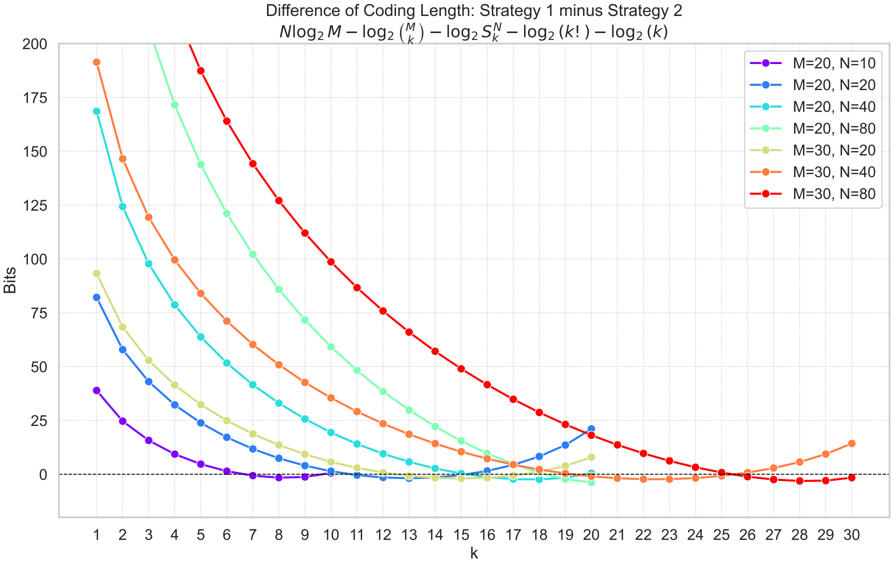

Figure 3: Illustration of Proposition 46. Using U_{compCBN}, the difference in coding length of strategy 1 minus strategy 2 is initially positive and decreases with k, then becomes negative. If there are very few mechanisms used, then U_{compCBN} will prefer an FCM that uses the least number of probabilistic mechanisms.

Remark 47 *Proposition 46 shows that in the trade-off of eq. (4.4), Strategy 1 can be preferred to Strategy 2 when the data is noisy, namely when the coding length of the codeword part* $-\log \mathbb{P}(x)$ *increases. The intuition behind this is that "identity" or "sparse mechanism shifts" are all approximations; if the model with sparse mechanism shifts fits the data sufficiently well, compression prefers not to code too many mechanisms. The experiments in § 6 also show this trade-off.*

Appendix G. Details of experiments

G.1. Covariate shifts

Consider a multi-env system with covariate shift: $\mathbb{P}(X, Y, E) = \mathbb{P}(X|E)\mathbb{P}(Y|X)\mathbb{P}(E)$. Suppose many CFMPs $(\alpha_l)_{l\in[L]}$ generate the same $\mathcal{P}$ and Φ and featurize them in the same way, as described in strategy 2 before Proposition 46. The question is: among these CFMPs, under the UFCC U_{CompCBN} in Proposition 46, given multi-env finite data, if we only consider the case "X causes Y", which CFMP should we select? In the current experiment, we generate 10 environments for (X, Y) supported in $\{0, 1, \ldots 17\}^2$, each having 10 iid samples. We make the shifts in $\mathbb{P}(X|E)$

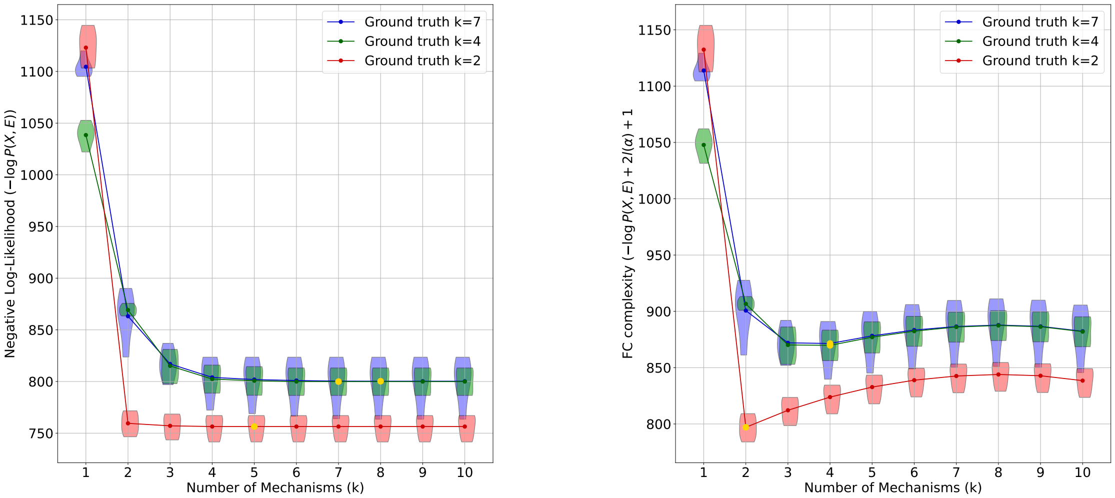

Figure 4: Results in § G.1. The left figure shows the minimal negative log-likelihood of the CFMPs that use k mechanisms $\mathbb{P}(X|E)$. The right figure shows the minimal FC complexity (NLL+model coding length $2l_{U_{\text{CompCBN}}}(\alpha) + 1$ (eq. (4.4))) of the CFMPs that use k mechanisms $\mathbb{P}(X|E)$. We choose 3 different multi-env distributions to generate the data, respectively using 2,4 and 7 mechanisms among 10 environments. The experiments are run with 5 seeds. The argmin k are highlighted.

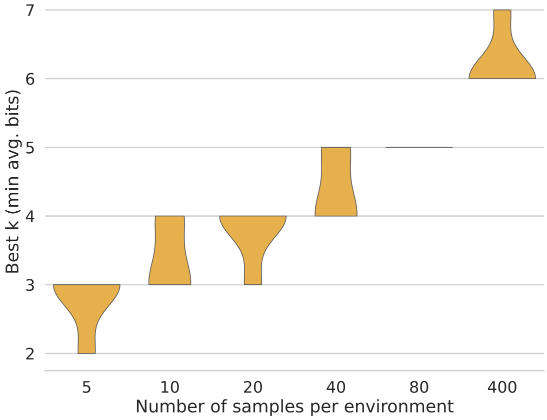

Figure 5: For the experiment in § G.1 with ground truth $k = 7$ for 10 envs, we increase the number of samples per env. As the number increases, the model selected by minimizing FC complexity tends to use more mechanisms in different environments.

sparse, i.e. less than 10, see the legends in Figure 4. $\mathbb{P}(X|E)$ is modeled by Poisson distribution, with the support outside of $\{0, 1, \ldots 17\}^2$ absorbed in $(X = 17)$. We define the invariant $P(Y|X) \sim \mathcal{N}(X, 1)$.

To proceed with model selection, we consider a set of CFMPs $(\alpha_l)_{l\in[L]}$ which have 18 featurized mechanisms for $\mathbb{P}(X|E)$, and the known $\mathbb{P}(Y|X)$ and $\mathbb{P}(E)$. Since the goal is to compare and select among these CFMPs, we do not calculate the coding length for steps 1 and 2 which is the same for all those CFMPs. The only step that makes a difference in coding length, step 3, costs $\log\binom{M}{k} + \log S_k^N + \log(k!) + \log k$ bits, as shown in Figure 3 and the proof of Proposition 46. Namely, the coding length in step 3 depends on how many different $\mathbb{P}(X|E)$ mechanisms α should use. Once k is chosen, α performs a greedy search over all possible choices of using k mechanisms in 10 environments and selects the choice that minimizes the negative log-likelihood (NLL) of multi-env data, which is equivalent to the sum of Shannon code length for the multi-env data.

The results in Figure 4 show that with finite data, if we only minimize NLL, we tend to select the ground truth mechanisms, and sometimes we choose even more mechanisms than the ground truth needs. However, by minimizing FC complexity (NLL+model length), we tend to trade off the NLL and the model complexity and select a simple (sparse mechanisms needed) model that explains the data well enough.

The parameters of Poisson distribution $\mathbb{P}(X|E)$ for each env are:

Blue curve ($k = 7$): $[0.1, 0.1, 0.1, 0.3, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9] \times 20$.

Green curve ($k = 4$): $[0.1, 0.1, 0.4, 0.4, 0.6, 0.6, 0.6, 0.9, 0.9, 0.9] \times 20$.

Red curve ($k = 2$): $[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.9, 0.9] \times 20$.

We set the list of Poisson parameters for any env to choose from:

$[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9] \times 20$.

For each $k \in [10]$, we use greedy search over all possible choices of k-many parameters, i.e. $\binom{18}{k}$. For each choice, we assign those k parameters to the environment that has the closest estimated mean, which is the estimated λ for Poisson distribution.

G.2. Causal discovery without identifiability

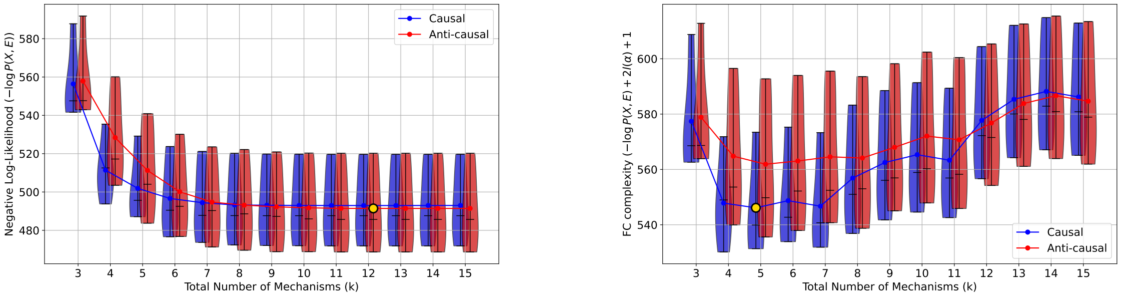

Figure 6: Results in § G.2. The x-axis is always the overall number of mechanisms (maximum 24 but truncated at 15). The left figure shows the minimal negative log-likelihood of the CFMPs that use k mechanisms. The right figure shows the minimal FC complexity (NLL+model coding length $2l_{U_{\mathrm{CompCBN}}}(\alpha) + 1$ (eq. (4.4))) of the CFMPs that use k mechanisms. The experiments are run with 5 seeds. The argmin k are highlighted.

Consider a multi-env system with 5 environments generated by linear Gaussian SCMs: $X \sim \mathcal{N}(0, \sigma_1^2), Y = aX + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_2^2)$. We generate 10 iid samples for each environment.

In total, 8 parameters are needed to fit the data optimally.

However, the causal graph is not identifiable, because we can also find parameters for each env using the linear Gaussian model $Y \to X$: $Y \sim \mathcal{N}(0, \tau_1^2), X = bY + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \tau_2^2)$.

For both causal and anti-causal linear Gaussian models, we allow 8 choices for each parameter, which include the optimal parameters, in total 24 parameters for the multi-env system. We constrain the number of mechanisms (in this case, parameters) before training the model, just as Strategy 2 in Proposition 18.

We see that FC complexity provides a new criterion of a "good model" that is applicable when the model is not identifiable from infinite sample and addresses the model selection problem, e.g., here fewer mechanisms than in the ground truth are selected.

The 5-env data is generated by the following Gaussian parameters $(\sigma_1^2, \sigma_2^2, a)$:

$$\begin{aligned} e = 0 :&[1, 16, 1], \\ e = 1 :&[1, 16, 1], \\ e = 2 :&[9, 16, 2], \\ e = 3 :&[9, 25, 2.5], \\ e = 4 :&[25, 25, 2.5]. \end{aligned}$$

For the causal model $Y = aX + \epsilon$, we select the parameters from a uniformly distributed set of 8 points spanning the range $[p_{\text{min}}, p_{\text{max}}]$ for each parameter $p \in \{(\sigma_1^2, \sigma_2^2, \gamma)\}$. For the anti-causal model $X = bY + \epsilon$, we compute the corresponding parameters in each env that achieve the same likelihood as the ground truth causal model, and create the set of candidate parameters in the same way as the causal model.

Appendix H. Remarks on Algorithmic Markov Condition (AMC)

Algorithmic Markov Condition (AMC) was introduced in Janzing and Schölkopf (2010). We first briefly recall their paper's main result, then compare the principles in it and in our paper. In the following, they use K to denote **prefix Kolmogorov complexity** (Li et al., 2019, Chapter 3).

Definition 48 (algorithmic Markov condition) *(Janzing and Schölkopf, 2010, Postulate 5) Let $x_1, \ldots, x_n$ be n strings representing descriptions of observations whose causal connections are formalized by a directed acyclic graph G with $x_1, \ldots, x_n$ as nodes. Let pa_j be the concatenation of all parents of x_j and nd_j the concatenation of all its non-descendants except x_j itself. Then*

$$x_j \perp nd_j \mid pa_j^*.$$

where $pa_j^* = (pa_j, K(pa_j))$, and the above independence is algorithmic, i.e. the algorithmic conditional mutual information $I(x_j : nd_j|pa_j^*) := K(x_j|pa_j^*) + K(nd_j|pa_j^*) - K(x_j : nd_j|pa_j^*) + O(1) = O(1)$.

Then they prove in their Thm. 3 that the above condition is equivalent to the statement

$$K(x_1, \ldots, x_n) = \sum_{j=1}^{n} K(x_j|pa_j^*) + O(1). \tag{H.1}$$

In other words, for n strings $x_1, \ldots, x_n$, if eq. (H.1) does not hold for a DAG G then we reject the statement that $x_1, \ldots, x_n$ satisfy an algorithmic causal model with DAG G, see their Postulate 6.

Inspired by eq. (H.1), they propose a principle for the DAG selection given a joint distribution P:

Principle 49 *(Janzing and Schölkopf, 2010, Postulate 7) A causal hypothesis G (i.e., a DAG) is only acceptable if the shortest description of the joint density P is given by a concatenation of the shortest description of the Markov kernels, i.e.*

$$K(P(X_1, \dots, X_n)) = \sum_j K(P(X_j|PA_j)) + O(1) \tag{H.2}$$

where $K(P)$ is the length of the shortest prefix-free program that computes $P(x, y)$ from (x, y). If no such causal graph exists, we reject every possible DAG and assume that there is a causal relation of a different type, e. g., a latent common cause, selection bias, or a cyclic causal structure.

We note that although eq. (H.2) seems similar to eq. (H.1), they are in fact different principles and one cannot derive one from the other. By Definition 39, the Kolmogorov complexity, whether prefix (K_U) or not (C_U), is a function that inputs strings instead of functions. A function has multiple string representations, so does a function or distribution P over $(X_1, \dots, X_n)$. If we consider P as n strings and apply eq. (H.1) to them, this is ill-defined because of the non-uniqueness of string representations. In fact, the motivation of proposing Principle 49 as a model selection principle is not related to eq. (H.1), but to what is afterwards named as "independent causal mechanism" (ICM) principle (Peters et al., 2017):

"We can think of $P(X)$ as describing a source S that generates x-values and sends them to a "machine" M that generates y-values according to $P(Y|X)$. Assume we observe that $I(P(X) : P(Y|X)) \gg 0$.[19] Then we conclude that there must be a causal link between S and M that goes beyond transferring x-values from S to M. This is because $P(X)$ and $P(Y|X)$ are inherent properties of S and M , respectively which do not depend on the current value of x that has been sent." (Janzing and Schölkopf, 2010, Sec. 3.1)

In other words, if the shortest Turing machines S and M computing respectively $P(X)$ and $P(Y|X)$ have algorithmic mutual information $O(1)$, then they accept the DAG $X \to Y$, otherwise reject.

Our principle of model selection, Principle 11, does not imply Principle 49. The reasons and comparisons are the following:

- Principle 11 allows selecting a CFMP that contains probabilistic mechanisms that are not algorithmically independent, for example $P(X) \sim \mathcal{N}(0, \sigma^2)$ and $P(Y|X) \sim \mathcal{N}(X, \sigma^2)$, which, according to Principle 49, leads to rejecting $X \to Y$ because of the compressibility of the shared parameter σ.

- Our output of model selection is different from Principle 49. We select a Turing machine, from which the causal and symmetry statements are read off. Principle 49 selects a graph only among all possible DAGs, which is less than the possible models that we illustrate in Example 1.

- The communication game setting behind Principle 49 is: Alice and Bob share a universal Turing machine, and Alice would like to send a string as short as possible to Bob so that Bob could compute a function P. In constrast, we are interested in the game where Alice would like to send *datasets*. The idea of UFCC is that Alice should send a codebook and a codeword so that Bob could reconstruct the datasets. A codebook might consists of a probability

[19] I denotes the algorithmic mutual information, see Definition 48.

distribution function P or not. Our choice of P depends on the trade-off between the two part codes, instead of being given a priori in eq. (H.2).

The last point shows that the two-part code objective in the sense of MDL or UFCC is fundamentally different from Principle 49. Marx and Vreeken (2021) aim at linking Principle 49 and two-part code in MDL. They propose a two-part code objective adapted from Budhathoki and Vreeken (2016):

$$K_{X\to Y} := K(P_X) + K(x|P_X) + K(P_{Y|X}) + K(y|x, P_{Y|X}) \tag{H.3}$$

and they proved that $K_{X\to Y}$ is on expectation equal to

$$K(P_X) + K(P_{Y|X}) + H(P_{XY}), \tag{H.4}$$

which is also a two-part code objective consisting of model length $K(P_X) + K(P_{Y|X})$ and the codeword length $H(P_{XY})$. Their two objectives eq. (H.3) and eq. (H.4), however, do not agree with $K(P_{XY})$ in Principle 49. Therefore, the two-part code objective (whether from MDL or UFCC) and Principle 49 cannot be derived from each other.

Appendix I. Remarks on Minimum Description Length principle (MDL) and Bayesian model selection

In Definition 13 we define that each codebook computed by a UFCC must be a function $\mathcal{X}^d \to \mathcal{B}^*$. When the data is iid sampled from a distribution over $\mathcal{X}^d$ then Huffman code is optimal (Theorem 37). If we modify the definition of UFCC by changing the domain of codebook from $\mathcal{X}^d$ to $(\mathcal{X}^d)^*$, and if the data is exchangeable instead of iid, then there is a code called Bayes code that has shorter codeword length than a given Huffman code for all sequence $x \in (\mathcal{X}^d)^*$. This is a fundamental idea in MDL principle (Grünwald, 2007):

Example 8 (Example 6.4 in Grünwald (2007): Bayes code is better than two-part code)
The Bayesian model is in a sense superior to the two-part code. Namely, in the two-part code we first encode an element in the parameter set Θ using some code L_0. Such a code must correspond to some "prior" distribution W on $\mathcal{M}$ so that the two-part code gives codelengths

$$L_{\textit{2-part}}(x^n) = \min_{\theta\in\Theta} -\log \mathbb{P}_\theta(x^n) - \log W(\theta), \tag{I.1}$$

where W depends on the specific code L_0 that was used.
Define the Bayes code with prior W:

$$L_{\textit{Bayes}}(x^n) := -\log \mathbb{P}_{\textit{Bayes}}(x^n) = -\log \sum_{\theta\in\Theta} \mathbb{P}_\theta(x^n) W(\theta) \tag{I.2}$$

where $P_{\textit{Bayes}}$ is the marginal likelihood of the data x^n under the prior W. Then it is direct to see that

$$L_{\textit{Bayes}}(x^n) = -\log \sum_{\theta\in\Theta} \mathbb{P}_\theta(x^n) W(\theta) \le \min_{\theta\in\Theta} -\log \mathbb{P}_\theta(x^n) - \log W(\theta) = L_{\textit{2-part}}(x^n)$$

because a sum is at least as large as each of its terms.

The inequality becomes strict whenever $\mathbb{P}_\theta(x^n) > 0$ for more than one value of θ. We see that in general the Bayesian code is preferable over the two-part code: for all x^n it never assigns code lengths larger than $L_{2-part}(x^n)$, and in many cases it assigns strictly shorter codelengths for some x^n.

The above example shows that for any two-part code there exists a Bayes code that is uniformly shorter than that two-part code. Therefore, the MDL research prefers Bayes code to two-part code, respectively termed refined MDL and crude MDL in Grünwald (2007).

Using this example, we can define a **Universal Bayes Codebook Computer (UBCC)**[20]:

Definition 50 *A Turing machine is called Universal Bayes Codebook Computer (UBCC) if it is constructed as follows:*

1. *First, same as in Definition 13, define any recursively enumerable set S of FCMs, such that any codebook $g : \mathcal{X}^d \to \mathcal{B}^*$ can be computed by at least one of the FCMs in it (In the following, we will call such a r.e. set a universal set of FCMs.).*
2. *Define a discrete probability $\mathbb{Q}$ fully supported over that countable set $\mathcal{S}$.*
3. *For any $x \in (\mathcal{X}^d)^*$, compute its marginal likelihood $\mathbb{P}(x) = \sum_{T \in \mathcal{S}} \mathbb{P}(x|T)\mathbb{Q}(T)$, where $\mathbb{P}(\cdot|T)$ is the probability distribution function computed by T. By Corollary 34, the negative log marginal likelihood $-\log \mathbb{P}$ is the coding length of a certain Shannon code over $(\mathcal{X}^d)^*$. By Example 8, this code is shorter than any two-part code in any UFCC using the same r.e. set $\mathcal{S}$ of FCMs.*

And different from FC complexity which is a two-part code, we define ***Bayes codebook complexity (BC)*** *$C_V^{BC}(\cdot)$ as a one-part code, i.e. the shortest integer $p \in \mathbb{N}$ such that the codebook corresponding to $-\log \mathbb{P}$ above can decode $B(p)$ and output x.*

In other words, a UBCC is one single infinite codebook mechanism, over $(\mathcal{X}^d)^*$. Given a UFCC V, we obtain a r.e. set $\mathcal{S}$ of FCMs, and we can build a UBCC V' using $\mathcal{S}$ by assigning each FCM $T \in \mathcal{S}$ to a prior probability $\mathbb{Q}(T) = 2^{-l_V(T)}$. By Example 8, $C_{V'}^{\mathrm{BC}}(x) \leq C_V^{\mathrm{FC}}(x)$ for all $x \in (\mathcal{X}^d)^*$. Namely, $C^{\mathrm{BC}_{V'}}$ is a more refined upper bound of Kolmogorov complexity than C^{FC_V}, although the r.e. sets of Turing machines that V and V' simulate are disjoint: V' always *computes* the same codebook and feed it with different binary inputs to output different x, while V *simulates* explicitly each FCM of which the input is a binary sequence.

However, to proceed the model selection over $\mathcal{S}$, Bayesian model selection in UBCC and two-part code model selection in UFCC coincide: they are both maximum a posteriori. For UFCC, the prefered FCM is $\min_{T,p}\{2l_V(T) + l(p)\}$ which equals $-\log \mathbb{Q}(T) - \log \mathbb{P}(x|T)$ for certain $\mathbb{P}$ and $\mathbb{Q}$ (the existence of them are again by Corollary 34). For UBCC, Bayesian model selection chooses the T that maximizes the posterior $\mathbb{P}(T|x)$.

The Bayesian model selection in Dhir et al. (2024) can be considered as a compression scheme between UFCC (pure two-part code) and UBCC (pure Bayes code). They defined their decision criterion of causal graph as the ratio of posterior

[20] Same as Definition 13 we omit (m, d) in the input for simplicity. In the general case we can construct UBCC depending on (m, d) by inputting $\langle m, \langle d, \langle p \rangle\rangle\rangle$ and simulating the codebook for each (m, d) respectively.

$$\log \frac{\mathbb{P}(G_1|x)}{\mathbb{P}(G_2|x)} = \log \frac{\mathbb{P}(x|G_1)\mathbb{P}(G_1)}{\mathbb{P}(x|G_2)\mathbb{P}(G_2)} \tag{I.3}$$

and G_1 is preferred to G_2 if the log ratio is positive. To represent the lack of knowledge over graph choices, they set the prior over graphs to be uniform. Since they choose the graph G that maximizes $\mathbb{P}(x|G)\mathbb{P}(G)$, which is equivalent to minimize $-\log \mathbb{P}(x|G) - \log \mathbb{P}(G)$, their objective is also a two-part code: first encode the graph, and then encode a *Bayes code* (negative log marginal likelihood) of x given G.

The main difference between Dhir et al. (2024) and the computational-theoretic objective (with reference machine UFCC or UBCC) is that the former approach (Dhir et al., 2024) aims at maximizing the posterior of a *graph*, while the latter aims at maximizing the posterior of a *Turing machine*, which, from Dhir et al. (2024) or a probabilistic point of view, determines a graph and conditional probabilities on it. For Dhir et al. (2024), the conditional probabilities in the selected model are uncertain under a given prior. The preference of theories on causality is, in our view, fundamentally subjective.

Here we summarize all the compression games we mentioned in our paper:

- Shannon source coding: as explained after Theorem 35, Alice and Bob know the data distribution $\mathbb{P}_X$. Alice wants to send iid samples losslessly to Bob using a codebook. Before sending iid data, they can design together a codebook.
  Question: what is the shortest expected average coding length for each sample, among all possible codebooks? (without counting the length of the codebook)

- Kolmogorov complexity: Alice and Bob share a programming language (C, Python) or a universal Turing machine. They do not know any structure of the data sequence x to be sent.
  Question: what is the length of the shortest program that Alice can send so that Bob can losslessly recover x?

- Finite codebook (FC) complexity (Definition 14): Alice and Bob share a finite universal codebook computer (UFCC, Definition 13). They know that the data sequence x is sampled in $\mathcal{X}^d$ with precision m. Before sending data, they can design together a codebook $\mathcal{X}^d \to \mathcal{B}^*$.
  Question: what is the length of the shortest program that Alice can send so that Bob can losslessly recover x? Since we define the UFCC to only accept two-part code (codeword and codebook), the question is equivalent to: what is the minimal sum of the length of the codewords and the length of the codebook mechanism (TM) to compress a certain sequence x?

- Bayes codebook (BC) complexity (Definition 50): Alice and Bob share a universal Bayes codebook computer (UBCC, Definition 50). They know that the data sequence x is sampled in $\mathcal{X}^d$ with precision m. Before sending data, they can design together a codebook $(\mathcal{X}^d)^* \to \mathcal{B}^*$.
  Question: what is the length of the shortest binary string (Bayes codeword) that Alice can send so that Bob can losslessly recover x?