PROVABLY TRACKING EQUIVALENT MECHANISTIC INTERPRETATIONS ACROSS NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mechanistic interpretability (MI) is an emerging framework for interpreting neural networks. Given a task and model, MI aims to discover a succinct algorithmic process, an **interpretation**, that explains the model's decision process on that task. However, MI is difficult to scale and generalize. This stems in part from two key challenges: the lack of a well-defined notion of a valid interpretation; and, the ad hoc nature of generating and searching for such explanations. In this paper, we address these challenges by formally defining and studying the problem of **interpretive equivalence**: determining whether two different models share a common interpretation, without requiring an explicit description of what that interpretation is. At the core of our approach, we propose and formalize the principle that two interpretations of a model are (approximately) equivalent if and only if all of their possible **implementations** are also (approximately) equivalent. We develop tractable algorithms to estimate interpretive equivalence and case study their use on Transformer-based models. To analyze our algorithms, we introduce necessary and sufficient conditions for interpretive equivalence grounded in the similarity of their neural representations. As a result, we provide the first theoretical guarantees that simultaneously relate a model's algorithmic interpretations, circuits, and representations. Our framework lays a foundation for the development of more rigorous evaluation methods of MI and automated, generalizable interpretation discovery methods.

1 INTRODUCTION

Ensuring the interpretability of deep neural networks has become central to concerns around AI safety and trustworthiness. Among the many proposed interpretation methods, **mechanistic interpretability (MI)** has recently emerged as a promising post-hoc interpretability framework[1] (Olah et al., 2020a; Elhage et al., 2021; Wang et al., 2022, *inter alia*). MI generally operates in two stages: **(a)** identifying a minimal subset of the model's computational graph that drives functional behavior; and **(b)** attaching algorithmic interpretations to each of the recovered mechanisms. In contrast to other attribution-based interpretability methods, MI yields a concrete, human-interpretable algorithmic process that faithfully describes model behavior (Bereska and Gavves, 2024). The resulting processes can be used to demystify training dynamics (Nanda et al., 2022) and better understand the model's inductive biases (Geva et al., 2021; Cabannes et al., 2024, *inter alia*) which further guide model improvements (Meng et al., 2022; McLeish et al., 2024).

For a fixed task and network, MI approaches can be broadly categorized as either **top-down** or **bottom-up** (Vilas et al., 2024). Top-down methods ($b \to a$) propose a set of high-level candidate algorithms for the task, and then attempt to align these algorithms to the network. While the alignment step can be automated and made statistically rigorous, proposing candidate algorithms is highly ad hoc. For complex tasks, it is often unclear how to even formulate plausible candidates. Additionally, alignment with a proposed algorithm is only a necessary condition for interpretability, and does not guarantee that the model truly implements the intended algorithm (Geiger et al., 2025; Wu et al., 2023; Geiger et al., 2024; Sun et al., 2025). In contrast, bottom-up methods ($a \to b$) first isolate mechanisms-of-interest within the model (called **circuits**) then assign interpretations to these circuits (Olah et al.,

[1] An interpretability method that does not require any retraining. These methods can be applied in parallel with inference and does not compromise the model's original performance.

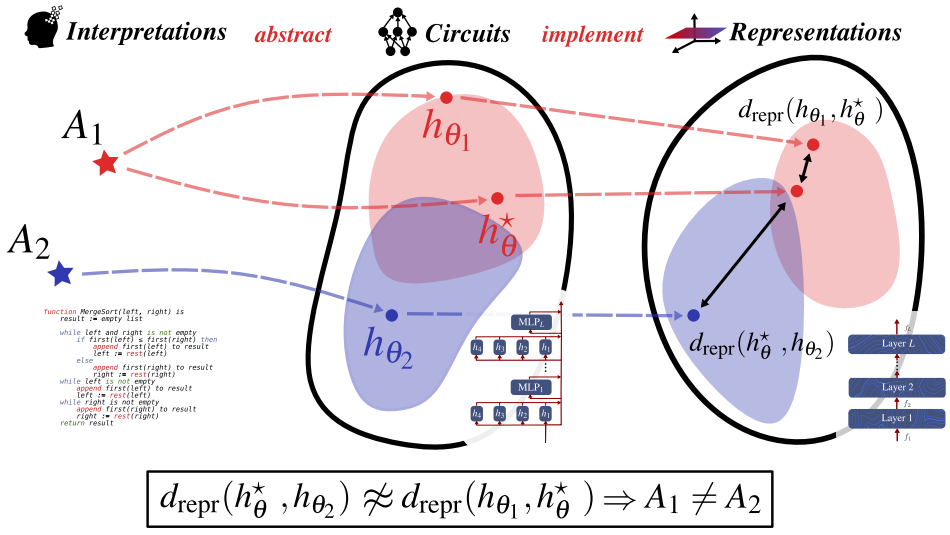

Figure 1: A high-level overview of our algorithmic approach to interpretive equivalence. Consider models $h_{\theta_1}, h_{\theta_2}$ that correspond to possibly *unknown* interpretations A_1, A_2 *(Left)*. To determine whether models h_{θ_1} and h_{θ_2} are interpretive equivalent, we propose a two-step procedure. First, we sample another model $h^\star_\theta$ that also has interpretation A_1 *(Center)*. Second, we compare the representation similarity (d_{repr}) between $h_{\theta_1}, h^\star_\theta$ and $h^\star_\theta, h_{\theta_2}$ *(Right)*. Intuitively, if models h_{θ_1} and h_{θ_2} are interpretive equivalent (i.e., $A_1 = A_2$), then averaged over all implementations $h^\star_\theta$, we should not be able to differentiate $d_{\text{repr}}(h_{\theta_1}, h^\star_\theta)$ and $d_{\text{repr}}(h^\star_\theta, h_{\theta_2})$.

2020a). Although circuit discovery can be rigorously formulated as an optimization problem, creating and assigning meaningful interpretations to circuits typically demands significant manual effort. Moreover, Méloux et al. (2025) has recently shown that neither top-down nor bottom-up approaches are identifiable: there exists a many-to-many relationship between high-level algorithms and circuits. This ambiguity makes it hard to verify whether a given interpretation is complete and faithful to the model (Jacovi and Goldberg, 2020; Chan et al., 2022). We present a detailed discussion of the related literature in Appendix C.

In this paper, we define and study a relaxed, subproblem of MI: **interpretive equivalence**. Concretely, we seek to determine whether two models implement the same high-level algorithm, **without** requiring an explicit description of what that algorithm is. Understanding interpretive equivalence can bridge the gap between bottom-up and top-down approaches by reconciling their respective limitations. We illustrate this through two examples:

Example 1.1 (Reduction to Simpler Models)**.** MI's scalability is limited by its prohibitive computational costs (Goldowsky-Dill et al., 2023; Adolfi et al., 2025). If a small model can be shown to be interpretively equivalent to a large one, then MI analyses on the small model can reveal the mechanisms underlying the larger model.

Example 1.2 (Reduction to Simpler Tasks)**.** MI's scope is limited by the significant human ingenuity required for both top-down and bottom-up approaches (Nanda et al., 2022; Zhong et al., 2023). For this reason, interpreting complex tasks is at least as hard as manually designing an algorithm to solve them. Interpretive equivalence offers a way to address this by decomposing complex tasks into simpler, approximate interpretive equivalent ones.

At the core of our approach (Figure 1), we propose the principle that two high-level algorithms (henceforth termed **interpretations**[2]) are equivalent if and only if all of their **implementations** are also equivalent. Based on this principle, we design tractable algorithms to detect equivalence by measuring representation similarity. We ground these contributions theoretically by proving that representation similarity is both sufficient and necessary to characterize interpretive equivalence. Overall, our contributions span both practice and theory:

[2]We distinguish between an algorithm and an interpretation. Colloquially, an algorithm is a description independent of any particular level of computational abstraction. However, we show in the coming sections that such algorithms are not well-defined. Thus, we opt for the terminology interpretation to signify a dependence on the level of computational abstraction.

Practice

- ***(P.i)*** We propose an algorithm to compute interpretive equivalence of two models through their representation similarity *without* interpreting them (Section 2 and Algorithm 1).
- ***(P.ii)*** We show that Algorithm 1 is well-calibrated on a simple task where the ground truth is known (Section 3.1).
- ***(P.iii)*** We demonstrate the potential of Algorithm 1 to find reductions from complex models and tasks to simpler ones as described in Examples 1.1 and 1.2 (Sections 3.2 and 3.3).

Theory

- ***(T.i)*** We specialize the theory of causal abstraction to define interpretations, circuits, representations, and interpretive equivalence (Sections 4 and 5).
- ***(T.ii)*** We prove that representation similarity is a necessary and sufficient approximation to interpretive equivalence, grounding our algorithmic contributions (Section 6).
- ***(T.iii)*** Other byproducts of our framework: a metric to quantify interpretation quality (Section 5), insight into interventions as a covering over implementation sets (Appendix G).

2 Interpretive Equivalence Through Ambiguous Representations

We define two mechanistic interpretations A_1, A_2 as equivalent if their implementations are equivalent. In other words, **any model that can be interpreted by A_1 must also by interpreted by A_2 and vice versa.** While we formally define and justify this principle later, we first argue its practicality and offer an algorithm to approximate this equivalence (Algorithm 1). To do so, we need to clarify two processes:

1. How do we enumerate the implementations of A_1, A_2? (Section 2.1)
2. How do we measure the distance between these sets of implementations? (Section 2.2)

2.1 Enumerating Implementations through Interventions

We call any model h that has a mechanistic interpretation A an **implementation** of A (Definition A.7). We take a circuits approach to generating implementations. Bottom-up approaches isolate circuits through targeted causal interventions. These interventions identify unimportant computational components like attention heads or even individual neurons that are unrelated to the model's behavior (Conmy et al., 2023; Goldowsky-Dill et al., 2023; Bhaskar et al., 2024). Crucially, a model's mechanistic interpretations are invariant under perturbation or ablation of these unimportant components. Dually, each time we find and ablate such a component, we yield a "new" model that is causally equivalent to the original one (and as a corollary, shares the same interpretation). We take this inverse perspective to generating implementations—adding, removing, or modifying these unimportant components (Geiger et al., 2024; Gupta et al., 2024). This procedure is described in GetImpl (Algorithm 1).

2.2 Representation Similarity

We identify each implementation with their hidden representation spaces (GetReprs in Algorithm 1). Given any two implementations, we use the linear representation similarity between them to measure their distance (d_{repr} defined in Definition F.1). Specifically, we measure the extent to which one representation can be reconstructed through a linear transformation of the other. Suppose that the implementation sets of A_1, A_2 were equal. Let $h_1, h_1^\star$ be implementations sampled[3] from A_1 and h_2 be an implementation sampled from A_2. By symmetry, $\mathbb{P}[d_{\text{repr}}(h_1, h_1^\star) < d_{\text{repr}}(h_2, h_1^\star)] = \mathbb{P}[d_{\text{repr}}(h_1, h_1^\star) > d_{\text{repr}}(h_2, h_1^\star)]$. In this way, the implementations of A_1, A_2 are "ambiguous" under d_{repr}. We present this metric as ReprDist (Algorithm 1). And, combining these concepts, we yield our approximation of interpretive equivalence: Ambiguity.

[3]For the sake of argument, suppose i.i.d. sampling.

Algorithm 1 Ambiguity between two models approximates the probability that two models are interpretive equivalent. Ambiguity between models is high when representation similarity *cannot* tell the difference between them, and low otherwise. $d_{\text{repr}}(h_{\theta_1}, h_{\theta_2})$ measures the linear representation similarity between the representations of $h_{\theta_1}, h_{\theta_2}$ (Definition F.1 and Sucholutsky et al. 2023). GETREPRS retrieves the hidden representations of a given model.

procedure AMBIGUITY($h_{\theta_1}, h_{\theta_2}, n$)
  s $\leftarrow 0$
  for $i \leftarrow 1 \ldots n$ **do**
    $h_{\theta_1^\star} \leftarrow$ GETIMPL(h_{θ_1})
    $h_{\theta_2^\star} \leftarrow$ GETIMPL(h_{θ_2})
    s $\leftarrow$ s + REPRDIST($h_{\theta_1}, h^\star_{\theta_1}, h_{\theta_2}$)
    s $\leftarrow$ s + REPRDIST($h_{\theta_2}, h^\star_{\theta_2}, h_{\theta_1}$)
  return $1 - |s/n - 1|$

procedure REPRDIST($h_{\theta_1}, h_{\theta_2}, h_{\theta_3}$)
  for $i \leftarrow 1, 2, 3$ **do**
    $R_i \leftarrow$ GETREPRS(h_{θ_i})
  if $d_{\text{repr}}(R_1, R_2) \leq d_{\text{repr}}(R_1, R_3)$ **then**
    return 1
  return 0

procedure GETIMPL(h_θ)
  $N \leftarrow$ components of h_θ whose ablation preserves performance
  $P \leftarrow$ components of h_θ whose ablation degrades performance
  Apply orthonormal transformations to a subset of P uniformly
  Perturb a subset of N uniformly with Gaussian noise
  Delete a subset of N uniformly
  Check that h_θ maintains performance
  return h_θ

3 EXPERIMENTS

Herein, we demonstrate three different applications of Algorithm 1. First, on a toy task where ground-truth interpretations are known, we show that AMBIGUITY is well-calibrated (Section 3.1). Next, we apply our framework to pre-trained language models of various sizes (GPT2 (Radford et al., 2019) and Pythia (Biderman et al., 2023)), and demonstrate that AMBIGUITY can distinguish between models that exhibit fine-grained algorithmic differences (Section 3.2). Lastly, we show how AMBIGUITY can be used to relate a complex task like next-token prediction to a simpler one such as parts-of-speech identification (Section 3.3).

3.1 CALIBRATING AMBIGUITY

We consider the task of n**-Permutation Detection**: determining whether a given sequence of n numbers is a permutation of the elements $1, \ldots, n$. For example, `[3, 1, 2]` $\rightarrow$ `True` and `[1, 2, 2]` $\rightarrow$ `False`. The task is sufficiently rich to support different interpretations, yet simple enough that solutions can be hard-coded as Transformers. We manually devise **six** different interpretations to solve this task whose procedures we detail in Appendix D. Roughly, their approaches can be organized into two buckets:

1. **Sorting-Based (Interpretations 1-4).** First sort the list of numbers by ascending (or descending) order, then directly check whether the resulting sequence is equal to $1, 2, \ldots, n$ (or $n, n-1, \ldots, 1$).
2. **Counting-Based (Interpretations 5-6).** Exploit the fact that the vocabulary contains exactly n numbers and use the pigeonhole principle to detect duplicates in the given sequence.

We fix $n = 10$. Using the Restricted Access Sequence Processing Language (RASP), we hard-code six bidirectional Transformers (of various architectures) to respectively implement each interpretation (Weiss et al., 2021). Then, for each hard-coded RASP Transformer, we leverage a technique of Gupta et al.'s (2024) to generate 100 model variants (with different architectures and weight configurations). These variants are constrained to maintain the same underlying interpretation as their hard-coded counterpart. Thus, we yield 6×100 models that all achieve 96%+ accuracy on 10-permutation detection. We now directly compute AMBIGUITY (Algorithm 1) between pairs of models from our generated implementation sets. We perform hypothesis testing by bootstrapping a 95% confidence interval over the final output[4]. The results are shown in Figure 2*(Left)*.

[4]Here, H_0 : the two models do not the same interpretation; and our alternate, H_1 : the two models share the same interpretations. We compute a Wald confidence interval with 20 straps.

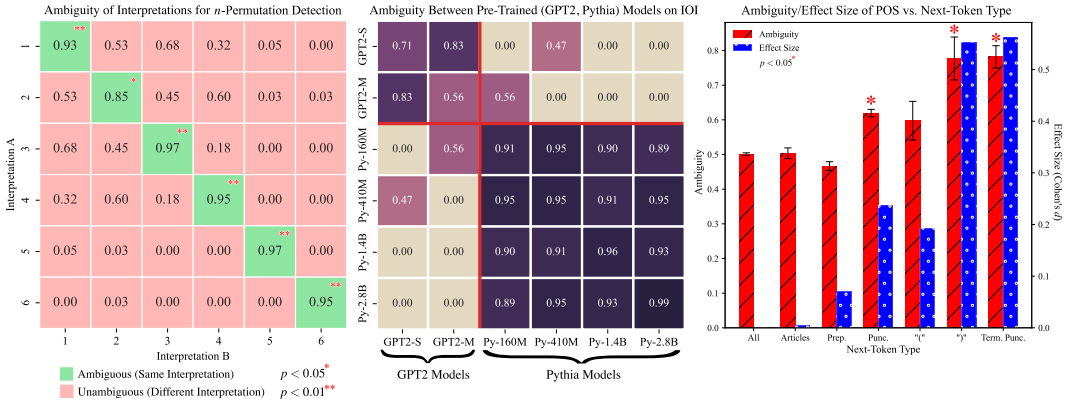

Figure 2: *(Left)* Average ambiguity between models associated with different interpretations. ■ indicates models have significantly different interpretations; whereas ■ indicates that models have statistically indistinguishable interpretations. *(Center)* Ambiguity between GPT2 and Pythia family of models on the IOI task. ▬ groups models based on their actual interpretive differences observed by Tigges et al. (2024); Merullo et al. (2024). *(Right)* Ambiguity between GPT2 on next-token prediction (for different token types: all tokens, articles, prepositions, punctuation, parentheses, and terminal punctuation) vs. GPT2 on in-context parts-of-speech identification.

Along the diagonals of Figure 2*(Left)*, we see significantly high ambiguity. So when models share the same interpretation, representations across their implementation sets are "ambiguous" (i.e. approximately equivalent under linear transformation). On the other hand, off-diagonal entries admit low ambiguity. Thus, representation similarity across implementation sets as a whole can identify individual interpretations. These empirical findings complement our bounds in Main Results 1 and 2 in that they show AMBIGUITY is both necessary and sufficient to detect interpretive equivalence. Perhaps even more interesting, we observe a breaking row/column in Figure 2, where Interpretations 1-4 (the 4×4 square on the top-left) have an average within-group ambiguity of 0.43. This is larger compared to the average across-group ambiguity (rows/columns 5-6) with Interpretations 5-6: 0.01. These groupings corresponds to our algorithmic buckets above and suggests that AMBIGUITY could also be adopted as a graded notion to characterize interpretive differences.

3.2 REDUCTION OF COMPLEX MODELS

We now consider the task of **indirect-object identification (IOI)** (Wang et al., 2022): given a sentence like "`When John and Mary went to the store, John gave a drink to` " the model should complete the sentence with "`Mary.`" IOI can be solved algorithmically, as such it has been studied across many models: GPT2-small/medium and the Pythia class of models. While Merullo et al. (2024) find that GPT2-small and medium use the same circuit, Tigges et al. (2024) find that Pythia models across all scales use a consistent but different circuit from GPT2 models (Merullo et al. 2024, Appendix C; Tigges et al. 2024: Appendix D). We use these differences as a practical testbed for AMBIGUITY and explore how AMBIGUITY generalizes across both model families and scales (85M to 2.9B parameters).

For each model, we generate 10 parallel implementations by intervening on the components found *not* to be in the IOI circuit (identified by both Tigges et al. 2024 and Merullo et al. 2024). Then, we apply AMBIGUITY, each time computing representation similarity with 200 IOI sentences. The results are shown in Figure 2*(Center)*. We find that the Pythia models show high within-group ambiguity across different scales. GPT2 models exhibit similar behavior. This supports our intuition that interpretive equivalence could be leveraged to reduce the interpretations of complex models into interpretations from smaller ones (Example 1.1). For example, our results affirm that Pythia-2.8b is interpretive equivalent to Pythia-160M with respect to the IOI task; thus, *a priori*, it suffices to only interpret the latter. Indeed, Tigges et al. (2024) finds both models to share the same interpretation. On average, across-group ambiguity between Pythia and GPT2 is significantly lower (0.13 versus 0.73 and 0.92) which supports our previous insight that representation similarity provides increased identifiability over interpretations. However, it is unclear why Pythia-160M and 410M show increased

ambiguity with GPT2-medium and small, respectively; perhaps, subtle similarities in name-mover heads representations could explain this (Tigges et al. 2024, Section 3.2).

3.3 Reduction of Complex Tasks

Interpreting next-token prediction poses a challenge for both bottom-up and top-down MI approaches. For top-down approaches writing down a symbolic, end-to-end algorithm for next-token prediction is difficult. Bottom-up approaches face the opposite problem: circuit discovery may identify the entire model as important, failing to reduce the search space of interpretations. We show here that interpretive equivalence may offer some first-steps towards mechanistically understanding next-token prediction. Concretely, we identify sets of next-tokens for which GPT2's prediction process is interpretive similar to parts-of-speech identification[5] (POS).

POS is a largely syntactic task since it focuses solely on the grammatical role of words rather than their meaning in-context. Thus, we expect POS to be interpretive equivalent to the prediction of "syntactic tokens." For computational ease, we conduct all experiments on GPT2. To discover the POS circuit in GPT2, we apply a method of Todd et al.'s (2024). We detail this process, along with the circuit we discover in Appendix D. We construct a dataset for next-token prediction by uniformly sampling 100 sentences from the C4 dataset (Raffel et al., 2020).

We collate tokens from our next-token prediction sentences into disjoint groups of interest. For each group, we apply Ambiguity and compute representation similarity between our extracted POS circuit and the last-token hidden representation of the model. The results across different token groups are shown in Figure 2*(Right)*.

As a control, we first consider the group of all tokens. We observe a nonzero ambiguity of 0.48 with POS. Next, we consider token groups **articles** and **prepositions**. Prediction of these tokens typically appear mid-sentence and depend heavily on semantics[6]. Thus, we expect that predicting these tokens should yield no more interpretive equivalence to POS than the control. Indeed, we find POS ambiguity for articles and prepositions to be statistically indistinguishable from the all-token average.

We now heuristically identify two sets of "syntactic tokens:"

1. **Terminal Punctuation** like ".", "?", or "!" mark the end of a sentence. Accurate prediction of these tokens intuitively requires syntactic identification of subject-verb-object relationships and (in)dependent clauses.
2. **Closing Brackets/Quotations** like ")" can be implemented as skip-trigrams in one-layer attention Transformers (Elhage et al., 2021). So, we suspect that understanding sentence pragmatics is not needed for their prediction.

We find that both groups yield significantly higher ambiguity compared our all-token control with medium effect size (measured through Cohen's d). This stands in contrast to punctuation generally and opening brackets/quotations. Although these groups admit significantly higher ambiguity with POS, we observe a small effect size ($d < 0.3$). We hypothesize that the placement of these tokens rely on more semantic processes. For example, commas may be placed for emphasis or to add dependent clauses rather than strictly maintain grammatical consistency.

4 Representations, Circuits, Interpretations as Causal Models

We now present our framework of interpretive equivalence. This theoretically grounds our algorithmic contributions in Section 2. Whenever possible, we present an informal treatment and defer the precise, mathematical definitions to Appendix A where we also have a full glossary. Please also refer to Appendix B for a summary of our notations.

[5] Given `tree:noun; run:verb; quick:adverb; fluffy:` the model needs to output `adjective`.

[6] Consider the fragment: "The students are looking `[mask]`" the next prepositions—"at", "into", or "for"—are all syntactically sound, but they are semantically ambiguous without further context. Articles are similar: consider, "I need to buy `[mask]` car." `[mask]` could be either the indefinite ("a") or definite ("the"), both syntactically valid but the choice depends on semantic factors.

Roadmap

1. In Section 4, we formally define **circuits** (Definition 4.2), **representations** (Definition 4.3), and **interpretations** (Definition 4.4).
2. In Section 5, we define a criterion for when two interpretations can be considered equivalent: **interpretive equivalence**. We also introdue **interpretive compression**, a metric quantifying the quality of an interpretation.
3. In Section 6, we prove our main results that representation similarity is both sufficient and necessary to describe interpretive equivalence (Main Results 1 and 2) and further show that AMBIGUITY is closely related to these quantities (Main Result 3).

Throughout the following sections, let $\Sigma^\star$ be all finite strings formed from alphabet Σ[7]. We define a **language model** to be a function $h_\theta : \Sigma^\star \to \mathbb{R}^d$, parameterized by $\theta \in \mathbb{R}^k$, for $d, k \in \mathbb{Z}^+$ Generally, we are interested in h_θ's behavior on a subset of inputs $S \subset \Sigma^\star$. Thus, we term S a **task**.

Definition 4.1 (Deterministic Causal Model, Geiger et al. 2025)**.** A causal model with m components is a quadruple $(\mathbb{V}, U, \mathbb{F}, \succ)$ where $\mathbb{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ is the set of **hidden** variables, U is an **input** variable, and $\succ$ a partial order of $\mathbb{V}$. $\mathbb{F} = \{f_1, \ldots, f_m\}$ is a set of functions where each f_k maps the values of $\boldsymbol{v}_k$'s parents (as determined by $\succ$) to the value of $\boldsymbol{v}_k$.

A causal model describes a computational graph where hidden variables $\mathbb{V}$ (nodes) store latent computation results computed by the functions in $\mathbb{F}$ (edges). For an input $U \leftarrow \boldsymbol{u}$, we denote $\mathbb{V}^{\mathbb{F}}(\boldsymbol{u})$ as the unique **solution**: the values of all hidden variables as determined by $\mathbb{F}$ (Peters et al., 2017). And, let $\boldsymbol{v}_i^{\mathbb{F}}(\boldsymbol{u})$ be the solution of $\boldsymbol{v}_i \in \mathbb{V}$. Next, we define **circuits**. For some task S, a circuit describes the computational pathways used by the model to produce $h_\theta(S)$.

Definition 4.2 (Circuit)**.** An m-circuit of h_θ on S is a causal graph $(\mathbb{V}, U, \mathbb{F}, \succ)$ with m components that satisfies: **(1)** U is S-valued; **(2)** Hidden variables are real-valued; **(3)** There exists a hidden variable $\boldsymbol{v}_{\text{out}}$ such that $\boldsymbol{v}_{\text{out}}^{\mathbb{F}}(S) = h_\theta(S)$.

There is a natural tradeoff between the complexity of $\mathbb{V}$ and the complexity of operators in $\mathbb{F}$. On one extreme, any blackbox model h_θ can be expressed as a **trivial causal model** with one hidden variable: $\boldsymbol{v}_1 \triangleq h_\theta(U)$. On the other, each $\boldsymbol{v} \in \mathbb{V}$ could be a single neuron (thus $|\mathbb{V}| \sim 10^9$), $\mathbb{F}$ consist of dot-products. Thus, intuitively, $\mathbb{V}$ determines the granularity of abstraction of the circuit. **Representations** of a neural network are sequence of activation spaces. Increasingly, representations have been understood to encode concepts which deep networks then iteratively refine (Jastrzebski et al., 2018). We view representations as abstrations of circuits and formally define them through causal abstraction (Beckers and Halpern, 2019; Geiger et al., 2025). Informally, causal model $K^\star$ **abstracts** K when there exists a surjective map between their variables that preserves K's causal relationships. We call any map that admits this abstraction an **alignment** from K to $K^\star$ (Definition A.4).

Definition 4.3 (Representations)**.** For a circuit K, an (L, δ)-representation of K abstracts K into a single chain of length L, where each hidden variable has exactly one parent. δ lower bounds approximation error in both directions: each hidden variable is rich enough to predict the output, yet simple enough that outputs can recover them (Equation A.3).

In MI, this latter constraint is often used as an empirical search criterion for "good representations" which in turn localizes a network's circuit (Belrose et al., 2025).

Interpretations are symbolic, human-understandable descriptions of h_θ that faithfully captures its functional behavior on a task. Formally, we view them as abstractions and do not attempt to define what makes them understandable, as this is inherently subjective.

Definition 4.4 (Interpretation)**.** Given a circuit K on S, an η-faithful interpretation is a causal model A that abstracts K such that A's output approximates K's with error at most η across all inputs in S.

We **do not assume** there exists a distance metric over A's variables nor do we specify the values of these variables take on. This generality aligns our framework with most MI literature.

[7] Our paper invokes practical examples based on language modeling. However, as we do not rely on properties of the input space, the theory we develop is domain agnostic and can be easily adapted to other modalities.

5 Interpretive Equivalence through Implementation Equivalence

Méloux et al.'s (2025) results demonstrate that pointwise comparison of circuits or interpretations is ill-defined, since many interpretations can correspond to a single circuit and vice versa. Thus, we propose that two interpretations are equivalent if and only if their implementations are equivalent. By examining families of circuits, we effectively quotient out the many-to-one mapping from circuits to interpretations. To start, let us define implementations. Let $K \triangleq (\mathbb{V}, U, \mathbb{F}, \succ)$ be an m-circuit of h_θ and $A \triangleq (\mathbb{V}^\star, U, \mathbb{F}^\star, \succ)$ be an η-faithful interpretation of K through an alignment π. For fixed π, by carefully varying $\mathbb{F}$ we could yield many different circuits over $(\mathbb{V}, U)$ that all abstract to A. **In our framework, $\mathbb{V}, U$ is the neural architecture and $\mathbb{F}$ are different weight configurations that yield the same interpretation A.** The set of all admissible $\mathbb{F}$ under A, π, we define as A's **implementations** under π. For a set of alignments, Π, we denote $\Pi^{-1}(A)$ as the union of all implementations under $\pi \in \Pi$. This construction closely mirrors the intuition of existing empirical results (Lubana et al., 2023; Zhong et al., 2023; Gupta et al., 2024).

We assume that any circuit comes with a metric attached to its variables, i.e. $d : \mathbb{V} \times \mathbb{V} \to [0, \infty)$. This is reasonable, as a circuit's hidden variables are real-valued. d then naturally induces a pseudometric over $\Pi^{-1}(A)$, where for $F, \tilde{F} \in \Pi^{-1}(A)$:

$$d(F, \tilde{F}) \triangleq d(\mathbb{V}^F, \mathbb{V}^{\tilde{F}}) = d(\mathbb{V}^F(S), \mathbb{V}^{F'}(S)). \tag{5.1}$$

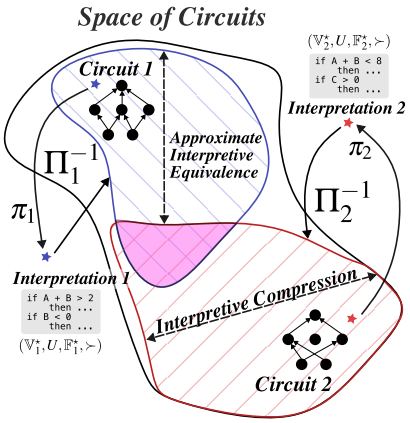

Figure 3

Given two sets of implementations over $(\mathbb{V}, U, \cdot, \cdot)$ we can leverage d to measure **(1)** the Hausdorff distance between them; and, **(2)** their diameters. We call the former **approximate interpretive equivalence** which we denote as $d_{\text{interp}}(A, A^\star)$ for interpretations $A, A^\star$, and the latter **interpretive compression** which we denote as $\kappa(A, K, \Pi)$ (Figure 3). The intersection between Π^{-1}, Π^{-2} then explains Méloux et al.'s (2025) non-identifiability of circuits and interpretations. In this way, interpretive equivalence and compression are tightly coupled. On one extreme, suppose $A = K$ and $|\Pi| = 1$. Then, there exists exactly one implementation of A under Π: K. This implies no interpretive compression. Equivalently, any interpretation equivalent to A must admit the same exact same weight configuration as K: $\mathbb{F}$. This effectively creates a bijective mapping between model weights and interpretations. On the other, suppose that A is the trivial causal model of h_θ, and let Π be all alignments. Since A conveys no information about the computational process of K and Π places no constraint on alignment, any circuit must be an implementation of A. In fact, Sutter et al. (2025) show under mild assumptions that this phenomenon persists as long as Π is the set of all alignments. In this case, interpretive compression is maximal. These examples illustrate a duality between equivalence and compression: while more compression shrinks and simplifies interpretation, it simultaneously enlarges and complicates implementation and equivalence. We further prove critical properties of interpretive equivalence and compression in Appendix E.

6 Representational Similarity and Interpretive Equivalence

Using interpretive compression and representation simiarlity, we now prove both upper and lower bounds on interpretive equivalence (Main Result 1 and Main Result 2, respectively). The proofs and formalisms are deferred to Appendix F. Throughout this section, we consider the most general setup: suppose two circuits over a shared task S for models h_1, h_2 defined over different neural architectures. Their circuits we notate as K_1, K_2, respectively. Let us also assume each circuit admits an (L_i, δ_i)-representation R_i through an alignment map ρ_i. First, we define a notion of distance between these two representations.

Definition 6.1 (Representation Similarity)**.** Let H_i be the concatenation of all hidden variables in R_i. Then, the representation similarity of R_1, R_2, $d_{\text{repr}}(R_1, R_2)$, is the bidirectional linear approximation error between H_1, H_2 (formally, Definition F.1).

Strict linearity in this approximation is not necessary to develop our theory, but offers attractive computational tradeoffs. This is because we can formulate representation similarity as multi-target linear regression for which there are sharp approximations with small sample complexity. Additionally, we make the following structural assumptions

1. ρ_i is Lipschitz continuous with Lipschitz constant $\text{Lip}(\rho_i)$
2. There exists $\Delta > 0$ such that $\|h_1 - h_2\| < \Delta$ over S

Assumption 1 stabilizes representations such that small changes in circuitry lead to proportionally bounded representation perturbations. In practice, this is reasonable since representations are usually distilled from neural networks via linear projections or by averaging the activations of several neural components (Sucholutsky et al., 2023). Assumption 2 guarantees that functional differences between our target models are bounded, otherwise their interpretive **in**equivalence is trivial.

Suppose that A_1, A_2 are η_1, η_2-faithful interpretations of K_1, K_2, respectively. We first show that representation similarity between A_1, A_2 upperbounds interpretive equivalence.

Main Result 1. *Let Π and $\Pi_\star$ be alignment classes that map K_1's variables into A_1, A_2's variables, respectively. Then, the approximate interpretive equivalence of A_1, A_2 under alignments $\Pi^{-1}, \Pi_\star^{-1}$ is*

$$d_{\text{interp}}(A_1, A_2) \lesssim \underbrace{\kappa(A_1, K_1, \Pi) + \kappa(A_2, K_1, \Pi_\star)}_{\text{(a)}} + \underbrace{\max(L_1\delta_1, L_2\delta_2) + d_{\text{repr}}(R_1, R_2)}_{\text{(b)}}. \tag{6.1}$$

Equation 6.1 explicits two dependencies: **(a)** measures interpretive compression of A_1, A_2 on K_1. When compression is low, circuits retain more information about interpretive differences. And since representations distill circuits, it is intuitive that representation similarity dominates the upperbound in this regime. In contrast, when compression is high, our examples in Section 5 illustrate that when circuits become uninformative, so should representations and their similarities; **(b)** jointly measures the quality of representations R_1, R_2 and their differences. If $\delta_1, \delta_2 \gg 1$, representations lose meaningful linear structure even though they sufficiently abstract implementations of A_1, A_2. This bound exposes a tension between the geometric richness of representations and our computational approach to their similarities. For example, representations lying on complex manifolds may not be well-described by our linear formulation of representation similarity (Ansuini et al., 2019; Pimentel et al., 2020). Thus, we hypothesize representations and their similarity criterion need to be simultaneously broadened to achieve more generality. We leave the implications of this for future work. Now, the lowerbound:

Main Result 2. *Let Π and $\Pi_\star$ be an alignment that map K_1's variables onto A_1, A_2's variables. Suppose A_1, A_2 are ε-approximately interpretive equivalent under the alignment classes $\Pi_\star, \Pi$. Then,*

$$d_{\text{repr}}(R_1, R_2) \lesssim \underbrace{\text{Lip}(\rho_1)(\min\{\kappa(A_1, K_1, \Pi), \kappa(A_2, K_1, \Pi_\star)\} + \varepsilon)}_{\text{(a)}} + \underbrace{L_1\delta_1 + L_2\delta_2}_{\text{(b)}} + \Delta. \tag{6.2}$$

The terms in Equation 6.2 mirror the dynamics of Equation 6.1. Beyond interpretive equivalence, Main Result 2 has implications for steering (Rimsky et al., 2024; Singh et al., 2024). When identifiability of representations is low, steering interventions is less likely to induce interpretive change, i.e. significant alterations to the representation space may see little change in model interpretation. In Main Result 3, we show how this directly relates to AMBIGUITY.

Main Result 3. *Let p be the expected value of* REPRDIST *in Algorithm 1, $F_1, F_2 \overset{iid}{\sim} \Pi_1^{-1}(A_1)$, and $F_3 \sim \Pi_2^{-1}(A_2)$ then*

$$p \geq 1 - \inf_{\varepsilon > 0} \mathbb{P}[d_{\text{repr}}(F_1, F_2) > \kappa(A_1, K_1, \Pi_1) + \varepsilon] - \mathbb{P}[d_{\text{repr}}(F_1, F_3) < d_{\text{interp}}(A_1, A_2) - \varepsilon]. \tag{6.3}$$

This variational lowerbound demonstrates that REPRDIST (and as a corollary, AMBIGUITY) accounts for interpretive compression and equivalence. We further discuss this bound in Appendix G.

7 CONCLUSION

In this paper, we define and study the problem of interpretive equivalence: detecting whether two models share the same mechanistic interpretation without interpreting them. We contribute an algorithm to estimate equivalence and demonstrate its use on toy and pre-trained language models. Then, we provide a theoretical foundation to ground and explain our algorithm. Our framework and results lay a foundation for the development of more rigorous evaluation methods in MI and automated, generalizable interpretation discovery methods.

8 Reproducibility Statement

We attach our complete codebase in the supplementary materials.

References

Federico Adolfi, Martina G. Vilas, and Todd Wareham. 2025. The computational complexity of circuit discovery for inner interpretability. In *The Thirteenth International Conference on Learning Representations*. 2, 17

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. 2019. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. 9

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in Language Models Is Mediated by a Single Direction. _eprint: 2406.11717. 18

Sander Beckers, Frederick Eberhardt, and Joseph Y. Halpern. 2020. Approximate causal abstractions. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 606–615. PMLR. 18

Sander Beckers and Joseph Y Halpern. 2019. Abstracting causal models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2678–2685. Issue: 01. 7, 16, 18

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. 2025. Eliciting latent predictions from transformers with the tuned lens. *Preprint*, arXiv:2303.08112. 7

Leonard Bereska and Stratis Gavves. 2024. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*. Survey Certification, Expert Certification. 1, 18

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding Transformer Circuits with Edge Pruning. *arXiv preprint arXiv:2406.16778*. 3, 17

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023. Emergent and predictable memorization in large language models. *Advances in Neural Information Processing Systems*, 36. 4

Vivien Cabannes, Charles Arnal, Wassim Bouaziz, Alice Yang, Francois Charton, and Julia Kempe. 2024. Iteration Head: A Mechanistic Study of Chain-of-Thought. In *Advances in Neural Information Processing Systems*, volume 37, pages 109101–109122. Curran Associates, Inc. 1

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldwosky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. 2022. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*. 2, 17, 18

Robin Chan, Reda Boumasmoud, Anej Svete, Yuxin Ren, Qipeng Guo, Zhijing Jin, Shauli Ravfogel, Mrinmaya Sachan, Bernhard Schölkopf, Mennatallah El-Assady, and Ryan Cotterell. 2024. On affine homotopy between language encoders. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 23

Arthur Conmy, Augustine Parker-Mavor N., Aengus Lynch, Stefan Heimersheim, and Adria Alonso-Garriga. 2023. Towards Automated Circuit Discovery for Mechanistic Interpretability. In *Thirty-Seventh Conference on Neural Information Processing Systems*. 3, 17, 19

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and others. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*. 19

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, DasSarma, Nova, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A Mathematical Framework for Transformer Circuits. 1, 6, 17

Dan Friedman, Alexander Wettig, and Danqi Chen. 2023. Learning Transformer Programs. In *Thirty-seventh Conference on Neural Information Processing Systems*. 19, 20

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. 2025. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64. 1, 7, 15

Atticus Geiger, Hanson Lu, Thomas F Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*. 16

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 2024. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236 of *Proceedings of Machine Learning Research*, pages 160–187. PMLR. 1, 3, 15, 18, 19, 20, 21

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 1

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. 2023. Looped Transformers as Programmable Computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR. 19

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching. *Preprint*, arXiv:2304.05969. 2, 3

Rohan Gupta, Iván Arcuschin, Thomas Kwa, and Adrià Garriga-Alonso. 2024. Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 3, 4, 8, 19, 20, 21

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*. 17

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have Faith in Faithfulness: Going Beyond Circuit Overlap When Finding Model Mechanisms. In *First Conference on Language Modeling*. 17, 19

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*. 16

Ehsan Imani, Wei Hu, and Martha White. Representation alignment in neural networks. *Transactions on Machine Learning Research*. 18

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics. 2

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*. 20

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. 2018. Residual connections encourage iterative inference. In *International Conference on Learning Representations*. 7

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of Neural Network Representations Revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR. 18, 23

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. 2024. A Mechanistic Understanding of Alignment Algorithms: A Case Study on DPO and Toxicity. In *Forty-first International Conference on Machine Learning*. 18

David Lindner, Janos Kramar, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. 2023. Tracr: Compiled Transformers as a Laboratory for Interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*. 19, 20

Ekdeep Singh Lubana, Eric J Bigelow, Robert P. Dick, David Krueger, and Hidenori Tanaka. 2023. Mechanistic Mode Connectivity. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22965–23004. PMLR. 8

Riccardo Massidda, Atticus Geiger, Thomas Icard, and Davide Bacciu. 2023. Causal Abstraction with Soft Interventions. In *Proceedings of the Second Conference on Causal Learning and Reasoning*, volume 213 of *Proceedings of Machine Learning Research*, pages 68–87. PMLR. 16, 18, 20

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and others. 2024. Transformers Can Do Arithmetic with the Right Embeddings. *arXiv preprint arXiv:2405.17399*. 1

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. 2025. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *The Thirteenth International Conference on Learning Representations*. 2, 8, 18

Kevin Meng, David Bau, Alex J. Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*. 1

William Merrill, Ashish Sabharwal, and Noah A. Smith. 2022. Saturated Transformers are Constant-Depth Threshold Circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856. _eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00493/2038506/tacl_a_00493.pdf. 19

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*. 5, 17

Lukas Muttenthaler, Lorenz Linhardt, Jonas Dippel, Robert A Vandermeulen, Katherine Hermann, Andrew Lampinen, and Simon Kornblith. 2023. Improving neural network representations using human similarity judgments. In *Advances in Neural Information Processing Systems*, volume 36, pages 50978–51007. Curran Associates, Inc. 18

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale. *URL: https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*. 17

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2022. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*. 1, 2, 18

Yaniv Nikankin, Anja Reusch, Aaron Mueller, and Yonatan Belinkov. 2025. Arithmetic without algorithms: Language models solve math with a bag of heuristics. In *The Thirteenth International Conference on Learning Representations*. 18

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020a. An overview of early vision in inceptionv1. *Distill*. Https://distill.pub/2020/circuits/early-vision. 1, 17

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020b. Zoom in: An introduction to circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in. 17

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html. 17

Jun Otsuka and Hayato Saigo. 2022. On the Equivalence of Causal Models: A Category-Theoretic Approach. In *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pages 634–646. PMLR. 18, 20

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The Linear Representation Hypothesis and the Geometry of Large Language Models. In *Forty-first International Conference on Machine Learning*. 15

Judea Pearl. 2009. *Causality*. Cambridge university press. 18

Jonas Peters, Dominik Janzing, and Bernhard Scholkopf. 2017. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press. 7, 18

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics. 9

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9. 4

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67. 6

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics. 9

Paul K. Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M. Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. 2017. Causal consistency of structural equation models. *Preprint*, arXiv:1707.00819. 22

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David Blei. 2024. Hypothesis testing the circuit hypothesis in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 17

Ravid Shwartz Ziv and Yann LeCun. 2024. To Compress or Not to Compress—Self-Supervised Learning and Information Theory: A Review. *Entropy*, 26(3). 18

Shashwat Singh, Shauli Ravfogel, Jonathan Herzig, Roee Aharoni, Ryan Cotterell, and Ponnurangam Kumaraguru. 2024. Representation surgery: Theory and practice of affine steering. In *Forty-first International Conference on Machine Learning*. 9

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*. 18

Ilia Sucholutsky, Lukas Muttenthaler, Adrian Weller, Andi Peng, Andreea Bobu, Been Kim, Bradley C Love, Erin Grant, Iris Groen, Jascha Achterberg, et al. 2023. Getting aligned on representational alignment. *arXiv preprint arXiv:2310.13018*. 4, 9

Alan Sun, Chiyu Ma, Kenneth Ge, and Soroush Vosoughi. 2024. Achieving domain-independent certified robustness via knowledge continuity. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 18

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. 2025. HyperDAS: Towards Automating Mechanistic Interpretability with Hypernetworks. In *The Thirteenth International Conference on Learning Representations*. 1, 18, 21

Denis Sutter, Julian Minder, Thomas Hofmann, and Tiago Pimentel. 2025. The non-linear representation dilemma: Is causal abstraction enough for mechanistic interpretability? *Preprint*, arXiv:2507.08802. 8

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*. 17

Hannes Thurnherr and Jérémy Scheurer. 2024. Tracrbench: Generating interpretability testbeds with large language models. *arXiv preprint arXiv:2409.13714*. 20

Curt Tigges, Michael Hanna, Qinan Yu, and Stella Biderman. 2024. LLM Circuit Analyses Are Consistent Across Training and Scale. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 5, 6

Naftali Tishby, Fernando C. Pereira, and William Bialek. 2000. The information bottleneck method. _eprint: physics/0004057. 18

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2024. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*. 6, 21

Mariya Toneva and Leila Wehbe. 2019. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. 18

Eduard Tulchinskii, Kristian Kuznetsov, Laida Kushnareva, Daniil Cherniavskii, Sergey Nikolenko, Evgeny Burnaev, Serguei Barannikov, and Irina Piontkovskaya. 2023. Intrinsic Dimension Estimation for Robust Detection of AI-Generated Texts. In *Advances in Neural Information Processing Systems*, volume 36, pages 39257–39276. Curran Associates, Inc. 18

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in neural information processing systems*, volume 33, pages 12388–12401. 16

Martina G. Vilas, Federico Adolfi, David Poeppel, and Gemma Roig. 2024. Position: An inner interpretability framework for AI inspired by lessons from cognitive neuroscience. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 49506–49522. PMLR. 1, 18

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations*. 1, 5, 16, 17

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR. 18

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking Like Transformers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR. 4, 19

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah Goodman. 2023. Interpretability at Scale: Identifying Causal Mechanisms in Alpaca. In *Advances in Neural Information Processing Systems*, volume 36, pages 78205–78226. Curran Associates, Inc. 1, 18

Aolin Xu and Maxim Raginsky. 2017. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in neural information processing systems*, volume 30. 18

Fabio Massimo Zennaro. 2022. Abstraction between structural causal models: A review of definitions and properties. *arXiv preprint arXiv:2207.08603*. 20

Zhiqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. 2023. The Clock and the Pizza: Two Stories in Mechanistic Explanation of Neural Networks. In *Advances in Neural Information Processing Systems*. 2, 8, 17, 18

A GLOSSARY

Herein, we provide a glossary for our technical terms in the main text along with their precise, formal definitions.

Term	Intuitive Explanation	Formal Definition
Causal/Computation Graph	A graph that describes a model's computational dependencies	Definition A.1
Variables	Nodes in a model's computation graph that store latent computation results	Definition A.1
Circuit	A subset of the model's computational graph that on its own can fully recreate the model's input-output behavior	Definition A.2
Mechanistic Explanation	A symbolic algorithm that we can understand that explains the model's computation	N/A
Causal Abstraction	Mapping from low-level variables to higher-level ones that preserve low-level causal relationships	Definition A.4
Interpretation	A causal graph much smaller than the model's computation graph that explains the model's computation graph	Definition A.6
Representations	A causal graph that is a chain that abstracts a model's circuit	Definition A.5
Alignment	A mapping between two causal graphs that preserves meaningful causal relations between variables	Definition A.4
Intervention	A modification to the model's computation graph	Definition A.3
Alignment Class	A set of alignments that map to the same interpretation	Definition A.7
Implementation	Sets of circuits that share the same interpretation	Definition A.7
Interpretive Equivalence	Measure of distance between sets of implementations	Definition A.9
Interpretive Compression	Measure of diameter of a set of implementations	Definition A.8
Ambiguity	Approximation for interpretive equivalence	Definition G.1
Representation Similarity	Extent to which one set of representations can be transformed into the other and vice versa	Definition F.1

Definition A.1 (Deterministic Causal Model, Geiger et al. 2025)**.** A (deterministic) causal model with m components is a quadruple $(\mathbb{V}, U, \mathbb{F}, \succ)$ where $\mathbb{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ is a set of **hidden** variables such that $|\mathbb{V}| = |\mathbb{F}| = m$, U is an **input** variable, and $\succ$ defines a partial ordering over $\mathbb{V}$. For each $\boldsymbol{v}_k \in \mathbb{V}$,

$$\boldsymbol{v}_k \triangleq f_k(Pa(\boldsymbol{v}_k), U)$$

where $f_k \in \mathbb{F}$ and $Pa(\boldsymbol{v}_k) \subset \{\boldsymbol{v} \in \mathbb{V} : \boldsymbol{v} \succ \boldsymbol{v}_k\}$ are the **parents** of $\boldsymbol{v}_k$.

Definition A.2 (Circuit)**.** An m-circuit of h_θ on S is a causal graph with m components that satisfies four properties:

1. Input variable is S-valued: $U \in S$
2. Hidden variables are real-valued: for each $\boldsymbol{v}_k \in \mathbb{V}$, $\boldsymbol{v} \in \mathbb{R}^{n_k}$
3. Existence of a terminal output variable: $\boldsymbol{v}_{\text{out}} \in \mathbb{V}$ such that $\boldsymbol{v}_{\text{out}} \in \mathbb{R}^d$, and there does not exist $\boldsymbol{v}' \in \mathbb{V}$ where $\boldsymbol{v}_{\text{out}} \succ \boldsymbol{v}'$
4. Sufficient description of h_θ on S: for each $s \in S$, $\boldsymbol{v}^{\mathbb{F}}_{\text{out}}(s) = h_\theta(s)$

For any h_θ and m, an m-circuit of h_θ must exist by the trivial causal model. m-circuits are also not unique. For instance, a joint change of bases to any $\boldsymbol{v}_k \in \mathbb{V} \setminus \boldsymbol{v}_{\text{out}}$ and f_k results in a new causal model that preserves functional faithfulness to h_θ (Park et al., 2024; Geiger et al., 2024).

Definition A.3 (Intervention)**.** Let $(\mathbb{V}, U, \mathbb{F}, \succ)$ be a causal model. An **intervention** of $\boldsymbol{v}_k \in \mathbb{V}$, $\text{do}(\boldsymbol{v}_k \leftarrow \tilde{f})$, redefines $\boldsymbol{v}_k \triangleq \tilde{f}(\tilde{Pa}(\boldsymbol{v}_k), U)$, for $\tilde{Pa}(\boldsymbol{v}_k) \subset \{\boldsymbol{v} \in \mathbb{V} : \boldsymbol{v} \succ \boldsymbol{v}_k\}$. We denote the set of interventions on $\boldsymbol{v}_k$ as $\mathscr{I}(\boldsymbol{v}_k)$.

In other words, we place the process using to derive $\boldsymbol{v}_k$'s value, f, with a new function $\tilde{f}$. In this paper, we exclusively focus on **hard** interventions, where $\tilde{f}$ is a constant function. This essentially ablates $\boldsymbol{v}_k$, as its values no longer depend on the input (Massidda et al., 2023).

Given any circuit, we can intervene on its hidden variables through **activation patching** (Vig et al., 2020; Wang et al., 2022; Heimersheim and Nanda, 2024): ablating a variable or modifying the process used to derive its value. Interventions serve as a check of consistency and equivalence between circuits and their interpretations (Beckers and Halpern, 2019; Geiger et al., 2021).

Definition A.4 (Abstraction)**.** Let $K_1 \triangleq (\mathbb{V}, U, \mathbb{F}, \succ_1), K_2 \triangleq (\tilde{\mathbb{V}}, \tilde{U}, \tilde{\mathbb{F}}, \succ_2)$ be causal models. K_2 **abstracts** K_1 if there exists surjective mappings $\pi : \text{SubsetsOf}(\mathbb{V}) \to \tilde{\mathbb{V}}, \omega : \mathscr{I}(\mathbb{V}) \to \mathscr{I}(\tilde{\mathbb{V}})$ that satisfies for all $\boldsymbol{v}_k \in \mathbb{V}, \tilde{\boldsymbol{v}}_k \in \tilde{\mathbb{V}}, \text{do}(\boldsymbol{v}_k \leftarrow f) \in \mathscr{I}(\boldsymbol{v}_k)$:

$$\tilde{\boldsymbol{v}}_k = \pi\left(\bigcup_{\boldsymbol{v}_k \in \pi^{-1}(\tilde{\boldsymbol{v}}_k)} f_k(Pa(\boldsymbol{v}_k), U)\right), \tag{A.1}$$

$$\pi(\text{do}(\boldsymbol{v}_k \leftarrow f)) = \text{do}(\pi(\boldsymbol{v}_k) \leftarrow \omega(f)). \tag{A.2}$$

We call π an **alignment**.

Equation A.1 describes an observational consistency constraint: if K_2 abstracts K_1, then by only observing the hidden variables of K_1 we can infer all hidden values of K_2. Viewed under this lens, the alignment π projects the fine-grained variables $\mathbb{V}$ to the coarse-grained ones $\tilde{\mathbb{V}}$. On the other hand, Equation A.2 is an intervention consistency constraint—stating that π and ω must commute. This implies that all of the causal relationships between variables in K_2 can be constructed by relationships between variables in K_1.

Definition A.5 (Representations)**.** Let K be an m-circuit of h_θ on a task S. An (L, δ)-representation of K, denoted by R, is an L-circuit that abstracts K such that for each $\boldsymbol{v}_k \in \mathbb{V}$

- $Pa(\boldsymbol{v}_{k+1}) = \{\boldsymbol{v}_k\}$ and $Pa(\boldsymbol{v}_1) = \emptyset$
- There exists linear maps $A_k : \mathbb{R}^{n_k} \to \mathbb{R}^d, B_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ where

$$\underbrace{\|A_k \boldsymbol{v}_k^{\mathbb{F}} - h\| < \delta}_{\text{signal}} \quad \text{and} \quad \underbrace{\|B_k h - \boldsymbol{v}_k^{\mathbb{F}}\| < \delta}_{\text{noise}} \tag{A.3}$$

  for some function norm (to be specified)

Definition A.6 (Interpretation)**.** For a model h_θ with m-circuit $K \triangleq (\mathbb{V}, U, \mathbb{F}, \succ)$ on a task S, an η-faithful interpretation of K is a causal model $A \triangleq (\mathbb{V}^\star, U, \mathbb{F}^\star, \succ)$ that satisfies two properties: **(1)** A abstracts K; **(2)** There exists a terminal output variable $\boldsymbol{v}_{\text{out}}^\star \in \mathbb{V}^\star$ such that $\|\boldsymbol{v}_{\text{out}}^{\star,\mathbb{F}} - \boldsymbol{v}_{\text{out}}^{\mathbb{F}}\| < \eta$ for the terminal output variable $\boldsymbol{v}_{\text{out}} \in \mathbb{V}$.

For any circuit, K is an interpretation of itself (under the identity abstraction), albeit not very useful.

Definition A.7 (Implementation)**.** Let Π be a class of alignments. Then, denote $\Pi^{-1}(A)$ the set of implementations of A where $F \in \Pi^{-1}(A)$ if A is an η-faithful interpretation of $(\mathbb{V}, U, F, \prec_F)$ under some alignment $(\pi : \text{SubsetsOf}(\mathbb{V}) \to \mathbb{V}^\star) \in \Pi$.

Definition A.8 (Compression)**.** Let A be an η-faithful interpretation of K and Π be a class of alignments. Then, the interpretive compression of A on K is

$$\kappa(A, K, \Pi) \triangleq \sup_{F, F' \in \Pi^{-1}(A)} d(F, F') = \text{diameter}(\Pi^{-1}(A)). \tag{A.4}$$

Definition A.9 (Equivalence)**.** Let A_1, A_2 be η_1, η_2-faithful interpretations of K and Π_1, Π_2 be classes of alignments. Then, A_1, A_2 are ε-approximately interpretive equivalent if

$$d_{\text{interp}}(A_1, A_2) \triangleq d_{\text{H}}\left(\Pi_1^{-1}(A_1), \Pi_2^{-1}(A_2)\right) < \varepsilon, \tag{A.5}$$

where d_{H} is the Hausdorff distance[8] defined by d (see Equation 5.1).

[8]The Hausdorff distance between two sets A, B is the greatest distance a point from A, B needs to travel to reach B, A, respectively: $\tilde{d}(a, B) \triangleq \inf_{b \in B} d(a, b)$, $d_{\text{H}}(A, B) \triangleq \max(\sup_{a \in A} \tilde{d}(a, B), \sup_{b \in B} \tilde{d}(b, A))$.

Definition A.9 requires that both interpretations, A_1, A_2, come from the same circuit K. This constraint can be relaxed by augmenting the alignment classes Π_i such that $(\pi : \text{SubsetsOf}(\mathbb{V}_1) \to \mathbb{V}_i^\star) \in \Pi_i$: essentially mapping the different interpretations onto the same neural architecture $\mathbb{V}_1$. Although this creates asymmetry in choosing which circuit to use for comparison ($\mathbb{V}_1$ or $\mathbb{V}_2$), **we should select the circuit of least compression that admits at least one implementation for both interpretations.** Our bounds in Section 6 justify this principle.

B Notations

Symbol	Meaning
Σ	An alphabet
$\Sigma^\star$	Set of all finite strings constructed from Σ
S	A subset of $\Sigma^\star$
A	An interpretation
K	A circuit
R	A representation
$\mathbb{V}$	Set of variables in a causal model
U	Input variable to causal model
$\mathbb{F}$	Functions in causal model that determine variable values
$\succ$	Partial ordering over variables of causal model induced by $\mathbb{F}$
$\mathbb{V}^{\mathbb{F}}(u)$	Solution of all variables to input u
$\boldsymbol{v}_k^{\mathbb{F}}(u)$	Solution of $\boldsymbol{v}_k$ to input u
π	An alignment between variables of two causal models
Π	A class of alignments
$\Pi^{-1}(A)$	The set of implementations of A induced by alignments Π
d_{interp}	Interpretive equivalence
d_{H}	Hausdorff distance
$\kappa(A, K, \Pi)$	Interpretive compression
d_{repr}	Representation similarity
$\text{Lip}(f)$	Lipschitz constant of f
$\overset{\text{iid}}{\sim}$	Sample i.i.d from some distribution

C Related Literature

There have been extensive studies in interpreting the decision criterion of neural networks (specifically, language models). With respect to our paper, they can be organized into the following categories.

Mechanistic Interpretability. Mechanistic Interpretability (MI) can be broadly broken into two distinct phases: first, identifying a minimal subset of the model's computational graph that is responsible for a specific behavior—this process is also referred to as *circuit discovery* Olah et al. (2020b;a); Conmy et al. (2023); Bhaskar et al. (2024); Hanna et al. (2024); second, assigning human-interpretable explanations to each of the extracted components—this process is generally referred to as *mechanistic interpretability* Chan et al. (2022); Wang et al. (2022); Zhong et al. (2023); Hanna et al. (2023); Merullo et al. (2024). This paper focuses on the latter process; thus, when we invoke the term mechanistic interpretability, we are referring specifically to this second phase. For completeness, we briefly review circuit discovery as well.

A model can be seen as a computational graph Elhage et al. (2021); Olsson et al. (2022): $G = (V, E)$ with vertices V and edges $E \subset V \times V$. A circuit is then any binary function $f : E \to \{0, 1\}$. The optimal circuit f satisfies two properties: (1) it minimizes $\sum_{e \in E} f(e)$; and (2) when edges not in the circuit $e \in E, f(e) = 0$ are ablated, the model's functional behavior remains unchanged. Although circuit discovery can be concretely formulated as an optimization problem, it is computationally intractable in its naive form Bhaskar et al. (2024); Adolfi et al. (2025). As a result, many relaxations and heuristics have been developed Conmy et al. (2023); Syed et al. (2023); Nanda (2023); Hanna et al. (2024); Bhaskar et al. (2024). Nevertheless, there exists a solution set that can be statistically verified Shi et al. (2024).

The process of mechanistic interpretability, which we focus on in this paper, presents different challenges. MI involves iteratively generating hypotheses about the interpretations for the interactions between different components, then testing those hypotheses through carefully crafted interventions (Chan et al., 2022; Geiger et al., 2024; Méloux et al., 2025; Sun et al., 2025, *inter alia*). MI methods can be further clustered into two approaches: *top-down* and *bottom-up*. Top-down approaches start by enumerating hypotheses about the possible algorithms the model could be implementing. Then, they iterate through these hypotheses and isolate those with the closest causal alignment Wu et al. (2023); Geiger et al. (2024); Bereska and Gavves (2024); Vilas et al. (2024); Sun et al. (2025). On the other hand, bottom-up approaches are data-driven. They seek to directly find algorithmic explanations of the model's mechanistic behaviors by analyzing activations or attention patterns Nanda et al. (2022); Zhong et al. (2023); Lee et al. (2024); Arditi et al. (2024); Nikankin et al. (2025).

Casual Abstraction. Causality, and more specifically causal abstraction, seeks to formally characterize when a symbolic explanation faithfully explains a data-generating process (in our application, this data-generating process would be a model) Pearl (2009); Peters et al. (2017); Beckers and Halpern (2019); Beckers et al. (2020); Otsuka and Saigo (2022). Causal abstraction as defined in Definition A.4 seeks to align "low-level" models (neural networks) with "high-level" models (symbolic explanations). These theoretical constructions have driven the development of many top-down mechanistic interpretability methods such as Wu et al. (2023); Geiger et al. (2024); Sun et al. (2025) which directly verify this condition. While causal abstraction provides a necessary condition for a valid interpretation, recent works such as Méloux et al. (2025) argue that it alone is insufficient to fully characterize what constitutes a valid interpretation. Practical procedures that stem from this theory also fail to address this. This highlights the need for additional theoretical frameworks to complement causal abstraction.

On the other hand, causal abstraction represents a *hard* equivalence: two models either *are* or *are not* causal abstractions of each other. The binary nature of this definition fails to capture our intuition that explanations can vary in quality or completeness. Some interpretations may better capture the model's behavior than others, or may only explain a subset of the model's functionality. The hard equivalence of causal abstraction makes it difficult to reason about these partial or imperfect explanations. As a result, framemworks such as Beckers et al. (2020); Massidda et al. (2023) soften this criterion by introducing distance metrics on the total settings of the "high-level" model. In practice, these approaches face two main drawbacks:

1. It is challenging to define meaningful distance metrics when high-level interpretations involve discrete symbols and symbolic reasoning Sun et al. (2024).

2. Comparing interpretations between models requires fully interpreting each model first, which forces a computationally expensive top-down analysis approach.

Our framework addresses these limitations by focusing on *implementations* rather than interpretations directly. Since neural network computations are fundamentally real-valued, we can leverage natural distance metrics in their computational space. Additionally, by analyzing families of implementations rather than requiring complete interpretations, we can compare models' interpretive similarity without the overhead of fully interpreting each model first.

Representation Similarity. Representation similarity in neural networks has emerged as a fundamental research area whcih addresses how internal representations correspond across different models, domains, and biological systems Kornblith et al. (2019).

Representations of deep neural networks have shown to exhibit an array of powerful properties. They have given insights into training dynamics and the generalization capabilities of models Tishby et al. (2000); Xu and Raginsky (2017); Shwartz Ziv and LeCun (2024) and also provide a tractable method to compare the inductive biases between different models Wang and Isola (2020); Imani et al.; Skean et al. (2025). These representations have also shown to admit rich geometric properties Tulchinskii et al. (2023). In this paper, we measure representation through a learned linear regression. These techniques have been used extensively to compare different neural architecture and even human-language model similarity Toneva and Wehbe (2019); Muttenthaler et al. (2023).

D EXPERIMENTAL DETAILS

In this section, we detail our experimental methods. We first review constructs of RASP Weiss et al. (2021). Then, we demonstrate that RASP provides an interface to craft interpretations, and through methods like Geiger et al. (2024); Gupta et al. (2024) we can enumerate implementations of these interpretations. Lastly, we describe our experimental hyperparameters.

D.1 RASP INTERPRETATIONS

Background on RASP Programs. The Restricted Access Sequence Programming (RASP) language is a functional programming model designed to capture the computational behavior of Transformer architdectures (Weiss et al., 2021). RASP programs have shown use in mechanistic interpretability both as an effective benchmarking tool for faithfulness (Conmy et al., 2023; Hanna et al., 2024) and as a method to develop "inherently" interpretable language models (Friedman et al., 2023). Another line of work uses it (and other similar methods) as a proof technique to reason about the Transformer architecture's generalizability on a host of tasks (Weiss et al., 2021; Merrill et al., 2022; Giannou et al., 2023). In this paper, we focus on RASP's applications in interpretability.

RASP programs operate on two primary types of variables: *s-ops*, representing the input sequence, and *selectors*, corresponding to attention matrices. These variables are manipulated through two fundamental instructions: elementwise operations and select-aggregate. *Elementwise operations* simulate computations performed by a multilayer perceptron (MLP), while *select-aggregate* combines token-level operations, modeling the functionality of attention heads.

Every RASP program is equipped with two global variables `tokens` and `indices`, essentially primitive *s-ops*. `tokens` maps strings into their token representations:

```
token("code") = ["c", "o", "d", "e"]
indices("code") = [0, 1, 2, 3]
```

On the other hand, `indices` map n-length strings into their indices. That is, a list of $[0, 1, \ldots, n-1]$. Elementwise operations can be computed through composition. That is,

```
(3 * indices)("code") = [0, 3, 6, 9]
(sin(indices)) = [sin(0), sin(1), sin(2), sin(3)]
```

Tokens and their indices can also be mixed through *selection matrices* which are represented through the *s-op select*. This operations captures the mechanism of the QK-matrix. It takes as input two sequences K, Q, representing keys and queries respectively, and a Boolean predicate p and returns a matrix S of size $|K| \times |Q|$ such that $S_{ij} = p(K_j, Q_i)$. Then, the OV-circuit can be computed through the *select-aggregate* operation, which performs an averaging over an arbitrary sequence with respect to the aforementioned selection matrix. For example,

$$\texttt{aggregate}\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 1 & 1 & 0\end{bmatrix}, [10 \quad 20 \quad 30]\right) = [10015]\,.$$

The previous example is directly lifted from Lindner et al. (2023).

Compiling RASP Programs. The power of RASP programming lies in its ability to translate any RASP program into a Transformer, a process known as *compilation*. As described in Lindner et al. (2023), this involves a two-stage approach. First, a computational graph is constructed by tracing the *s-ops* in the program, identifying how these operations interact with and modify the residual stream. Elementwise operations are converted into MLP weights, and individual components are heuristically assigned to Transformer layers. For further details, we refer the reader to Lindner et al. (2023).

As observed by Lindner et al. (2023), this compilation through "translation" introduces inefficiencies. Specifically, the heuristic layer-assignment of RASP components results in Transformers that often contain more layers than they need to have. Moreover, since RASP enforces the use of categorical sequences and hard attention (we only allow Boolean predicates) it requires various *s-ops* to lie orthogonal to each other after embedding as Transformer weights. As a result, this leads to a much larger embedding dimension that is usually observed in actual Transformers (Elhage et al., 2022).

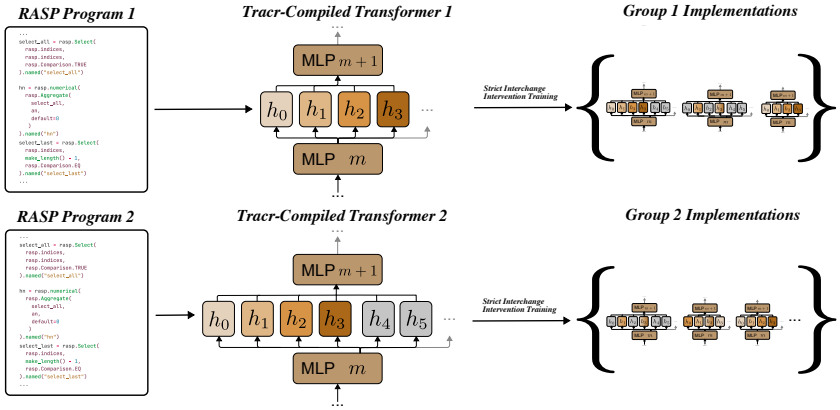

Figure 4: Pipeline for generating implementations for our interpretations (i.e. RASP language models). We first construct RASP programs, then using the procedure introduced in Lindner et al. (2023), we compile these programs into Transformer models. These Transformers exclusively have hard attention. Moreover, their architecture is minimal (containing only the necessary components to fully implement the given RASP program). We then apply the procedure introduced in Gupta et al. (2024) to translate these Tracr-Compiled Transformers into "real" Transformers: ones whose weight distribution matches those trained with stochastic gradient descent; these translated models also contain more

Thus, Lindner et al. (2023) proposes to compress this dimension through a learned projection matrix. The caveat is that this transformation largely not faithful to the original program (measured through cosine similarity of the outputs at individual layers).

Friedman et al. (2023) takes a different approach, addressing the inherent difficulty of writing RASP programs. To overcome this challenge, the authors propose a method for directly learning RASP programs. This is achieved by constraining the space of learnable weights to those that compile into valid RASP programs, ensuring outputs with categorical variables and hard attention mechanisms. Optimizing over this constrained hypothesis class is performed through a continuous relaxation using the Gumbel distribution (Jang et al., 2017).

RASP Benchmarks. Thurnherr and Scheurer (2024) is a dataset of RASP programs that have been generated by GPT-4. It contains 121 RASP programs. Gupta et al. (2024) provides 86 RASP programs and compiled Transformers. The compiled Transformers are claimed to be more realistic than Tracr compiled ones as instead of performing compression using a linear projection, they leverage *strict interchange intervention training* essentially aligning the intervention effects of the compressed and uncompressed model. This is similar in vein to many existing techniques on causal abstraction Otsuka and Saigo (2022); Zennaro (2022); Massidda et al. (2023). In our paper, we leverage the curated dataset Gupta et al. (2024) to craft and compose our interpretations.

D.2 Strict Interchange Intervention Training

To evaluate our methods, we need a way to verifiably ellicit different mechanisms on the same task. Let us first fix some task. Then, we proceed with the following steps:

1. Using Friedman et al. (2023), we learn several different explicit Transformer programs (source of randomness). We can check that they are different by looking explicitly at the Transformer programs.
2. Using Gupta et al. (2024) and Geiger et al. (2024) to get different mechanistic realizations of this abstract Transformer program.

To generate different mechanistic instantiations of the same interpretation across architectures, we use the following procedure: First, we take a Tracr-Compiled Transformer model and initialize a new random model with at least as many layers and attention heads. Two models share the same interpretation if the Tracr-Compiled transformer is a causal abstraction of our mechanistic instantiation. We leverage this insight by softening Definition A.4 and incorporating it directly into

our objective function. For a detailed treatment of this approach, we refer readers to Geiger et al. (2024); Gupta et al. (2024); Sun et al. (2025).

To ensure diversity in our implementations, we vary the architecture hyperparameters significantly: models contain between 2-6 layers, 2-8 attention heads, and embedding dimensions ranging from 32 to 2048. This creates a rich set of instantiations with widely varying model capacities. As a result, many mechanisms in these larger models may not contribute to the core interpretations, leading to substantial interpretive compression.

D.3 Algorithms for Permutation Detection

To implement this task, we proceed with the following six algorithms. As discussed in Section 3.1, they can be roughly divided into two groups: sort- and counting-based.

Interpretation 1. (1) Sort the sequence; (2) Compute the difference between each element and the next one; (3) Check if all elements of the sequence are equal.

Interpretation 2. (1) Sort the sequence in descending order; (2) Increment each element by its index; (3) Check if all elements of the sequence are equal.

Interpretation 3. (1) Sort the sequence; (2) Interleave the list with the same list in reverse order; (3) Sum each number with the number next to it; (4) Check if all elements of the list are equal.

Interpretation 4. (1) Sort the sequence; (2) Check if the list contains alternating even and odd elements.

Interpretation 5. (1) Check if at least two elements in the list are equal; (2) Sum each element with the next one in the list; (3) Check if all elements of the list are equal.

Interpretation 6. (1) Replace each element with the number of elements less than it in the sequence; (2) Check if at least two numbers are the same.

For each of the subroutines of these interpretations, Gupta et al. (2024) implements RASP programs for them. Thus, we simply compose them together to create the resulting models.

D.4 Function Vectors and Parts-of-Speech Identification

Function vectors have been found to drive in-context learning behavior (Todd et al., 2024). We leverage this concept to find the circuit that is responsible for in-context parts-of-speech identification. We use the Penn TreeBank dataset and consider the POS subtask. We sample $n = 1000$ points.

We create counterfactual inputs by shuffling the in-context labels. For example,

Clean Input: `tree:noun, run:verb, quickly:adverb, ire:`
Corrupted Input: `tree:adverb, run:noun, quickly:verb, ire:`

Then, we first run the model on the clean input and store all attention head output activations on the token "ire." We then run the model on the corrupted input and patch in the stored attention head activations from the "ire" token. Finally, we compute the difference in logit difference loss of the generated next tokens before/after this patching operation. The results for each attention head are shown in Figure 5. It seems that attention heads in the early/middle layers are most important for POS (0.1, 0.4, 0.6, 4.11, 5.1, 5.5, 6.8, 6.9, 6.10, 7.11).

Then, we compute the function vector by

$$\frac{1}{|A|}\sum_{a\in A}\sum_{k=1}^{n} a(x_k), \tag{D.1}$$

where A is the set of attention heads that we have deemed important and x_k is our dataset. Essentially, we are averaging the activations across all important attention heads.

E Properties of Interpretive Equivalence and Compression

As discussed in Section 5, an interpretation of a model need not be a lossless description of that model. In fact, in many applications a human-interpretable explanation is necessarily lossy with

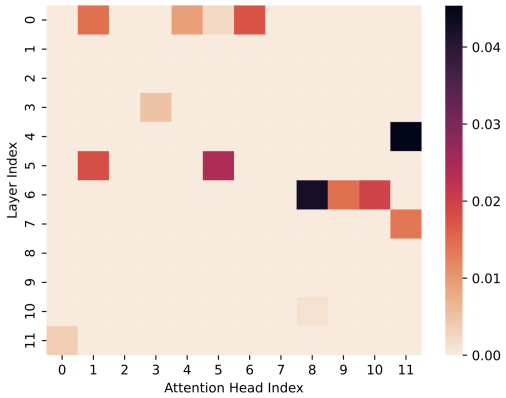

Figure 5: Activation patching results for each attention head. The coloring indicates the % of performance this individual attention head contributes to POS performance.

respect to the learned model. In this sense, a constructive abstraction is too restrictive as it requires any interpretation to preserve the exact functional behavior of the model.

Definition (Exact Transformation)**.** Let $M \triangleq (\mathbb{V}, U, \mathbb{F}, \succ), M_\star \triangleq (\mathbb{V}_\star, U, \mathbb{F}_\star, \succ)$ be two causal models. An exact transformation from M to $M_\star$ is a pair of partial surjective maps (τ, ω) where $\tau : \mathbb{V} \to \mathbb{V}_\star$ and $\omega : \mathscr{I}(M) \to \mathscr{I}(M_\star)$, an order-preserving map that satisfies Equation A.1 and Equation A.2.

The key property of an exact transformation is this commutativity.

E.1 Technical Lemmata

Lemma E.1 (Lemma 5 Rubenstein et al. 2017)**.** *The composition of exact transformations is also an exact transformation.*

Lemma E.2. *The composition of alignments is also an alignment.*

It directly follows from Lemma E.1 and Lemma E.2 that the composition of abstractions is also an abstraction. We shall use this fact next.

E.2 Proof of Basic Causal Implementation Properties

Lemma E.3. *Let K be a circuit and A be an η-faithful interpretation of K. Suppose that $A_\star$ is an $\eta_\star$-faithful interpretation of A. Then, $A_\star$ is an $(\eta + \eta_\star)$-interpretation of K.*

Proof. By Lemma E.2 and Lemma E.1, we know that there must exist an alignment from K to $A_\star$. It remains to show that this is a faithful one. Suppose that $\boldsymbol{v}_m, \boldsymbol{v}_{\text{out}}, \boldsymbol{v}^\star_{\text{out}}$ are the terminal output variables of $K, A, A_\star$, respectively. By faithfulness, it must be that

$$\begin{aligned} \|\boldsymbol{v}_m - \boldsymbol{v}^\star_{\text{out}}\| &\leq \|\boldsymbol{v}_m - \boldsymbol{v}_{\text{out}} + \boldsymbol{v}_{\text{out}} - \boldsymbol{v}^\star_{\text{out}}\|, \\ &\leq \|\boldsymbol{v}_m - \boldsymbol{v}_{\text{out}}\| + \|\boldsymbol{v}_{\text{out}} - \boldsymbol{v}^\star_{\text{out}}\|, \\ &\leq \eta_1 + \eta_2. \end{aligned}$$

□

Corollary E.4. *If $M_\star$ is an implementation of M and M is an implementation of A, then $M_\star$ must be an implementation of A.*

F Proofs for Representation Similarity and Interpretive Equivalence

Suppose we have two circuits over a shared task S for models $h_{\theta_1}, h_{\theta_2}$ defined over different neural architectures. These circuits we notate as $K_1 \triangleq (\mathbb{V}_1, U, \mathbb{F}_1, \succ)$ and $K_2 \triangleq (\mathbb{V}_2, U, \mathbb{F}_2, \succ)$, respectively.

Let us also assume each circuit admits an (L_i, δ_i) representation $R_i \triangleq (\mathbb{H}_i, U, \mathbb{F}_i, \succ)$ through an alignment map ρ_i. First, we define a notion of distance between these two representations.

Definition F.1 (Representation Similarity)**.** Denote by $H_i \triangleq \text{concat}\,(\mathbb{H}_i^{\mathbb{F}_i})$ to be the direct sum (concatenation) of all hidden variables in $\mathbb{H}_i$. Then, the representation similarity[9] between R_1 and R_2 is

$$d_{\text{repr}}(R_1, R_2) \triangleq \max(\inf_A \|AH_1 - H_2\|, \inf_B \|H_1 - BH_2\|), \tag{F.1}$$

where A, B are linear projections and $\|\cdot\|$ is some norm (to be specified, see Definition A.5).

F.1 Proof for Main Result 1

Definition F.2. Let (Z, d) be a metric space. Let $S \subset Z$. The diameter of S denoted diameter(S) is defined as

$$\text{diameter}(S) := \sup_{a,b \in S} d(a, b). \tag{F.2}$$

Lemma F.1. *Let (Z, d) be a metric space and d_{H} be its Hausdorff distance. Then, for $A, B \subset Z$ and $a \in A, b \in B$,*

$$d_{\text{H}}(A, B) \leq \text{diameter}(A) + \text{diameter}(B) + d(a, b).$$

Proof. Fix any $a' \in A$. Then,

$$\begin{aligned} \inf_{b' \in B} d(a', b') &\leq d(a', b), \\ &\leq d(a', a) + d(a, b), \\ &\leq \text{diameter}(A) + d(a, b). \end{aligned}$$

By symmetry, for any $b' \in A$, it follows that $\inf_{a' \in A} d(a', b') \leq \text{diameter}(B) + d(a, b)$. Therefore, by the definition of the Hausdorff distance, we yield the desired bound. □

Proof. By Lemma F.1, it suffices to bound the distance between any two implementations of K_1, K_2 in $\Pi^{-1}(A_1), \Pi_\star^{-1}(A_2)$. To do this, we directly construct valid circuits of I_1, I_2 using their representations and then bound their distance in d. Without the loss of generality, we manipulate the direct sum of the representations R_1, R_2: H_1, H_2. By construction, R_i takes on values in $\mathbb{R}^{\dim(H_i)}$.

If the representation similarity between R_1, R_2 is not finite, then the result holds trivially. Otherwise, consider the case where representation similarity between R_1, R_2 is bounded (equal to ε). Then, by assumption and Definition F.1, for some $\eta > 0$, there must exists a linear map $A : \dim(H_1) \to \dim(H_2)$ such that $\|AH_1 - H_2\| < \varepsilon + L_2\delta_2 + \eta$. Notice that R_1 must be a δ-faithful interpretation of A_1. Let B be the linear map such that $\|BH_2 - H_1\| < \varepsilon + L_1\delta_1 + \eta$. By our derivations in the proof of Theorem 2, AR_1 is also a $(L_1\delta_1 + \|B\|_{\text{op}}\varepsilon)$-faithful interpretation of A_1. It follows by triangle inequality and symmetry that

$$d(AR_1, R_2) \leq \max(L_1\delta_1, L_2\delta_2) + \varepsilon(\|B\|_{\text{op}} + \|A\|_{\text{op}}),$$

this yields the desired bound.

□

F.2 Proof for Main Result 2

Proof. Consider first an arbitrary $F \in \Pi_\star^{-1}(A_2)$. Then, since Definition F.1 satisfies the triangle inequality,

$$d_{\text{repr}}(R_1, R_2) \leq \underbrace{d_{\text{repr}}(R_1, R_2^\star)}_{\text{(a)}} + \underbrace{d_{\text{repr}}(R_2^\star, R_2)}_{\text{(b)}},$$

where $R_2^\star$ is the abstraction of $(\mathbb{V}_1, U, F, \succ)$ as a result of the alignment ρ_1. We now proceed to upperbound (a) and (b) separately.

[9] Our formulation is inspired by Chan et al.'s (2024) notion of representation alignment between language encoders. Although Definition F.1 is not a true similarity, this nomenclature stays consisten with the literature (Kornblith et al., 2019).

(a) Since A_1, A_2 are ε-approximately interpretive equivalent, it must be that

$$d(K_2^\star, K_1) \leq \varepsilon + \min(\kappa(A_1, K_1, \Pi), \kappa(A_2, K_1, \Pi_\star)).$$

Then, since ρ_1 is Lipschitz continuous, it follows directly that

$$d_{\text{repr}}(R_1, R_2^\star) \leq \text{Lip}(\rho_1)\left(\varepsilon + \min(\kappa(A_1, K_1, \Pi), \kappa(A_2, K_1, \Pi_\star))\right).$$

(b) Let $H_2^\star, H_2$ be the direct sum of the hidden variables in $R_2^\star$ and R_2, respectively. Then, by the definition of d_{repr},

$$d_{\text{repr}}(R_2^\star, R_2) = \max(\inf_A \|AH_2^\star - H_2\|, \inf_B \|H_2 - BH_2^\star\|),$$

for linear operators A, B. Again, we choose to bound both operands in the maxima separately. By assumption since R_1, R_2 are $(L_1, \delta_1), (L_2, \delta_2)$-representations, there must exist linear operators A_1, B_1, A_2, B_2 such that

$$\left\|A_1 H_1 - h_{\theta_1}\right\| < \delta_1 \qquad \left\|H_1 - B_1 h_{\theta_1}\right\| < \delta_1 \qquad \left\|A_2 H_2 - h_{\theta_2}\right\| < \delta_2 \qquad \left\|H_2 - B_2 h_{\theta_2}\right\| < \delta_2.$$

Then,

$$\begin{aligned}\inf_A \|AH_2 - H_2^\star\| &\leq \left\|B_2 A_1 H_2 - B_2 h_{\theta_2}\right\|, \\ &\leq \left\|B_2 A_1 H_2 - B_2 h_{\theta_2}\right\| + \left\|B_2 h_{\theta_2} - H_2\right\|, \\ &\leq \|B_2\|_{\text{op}} \left\|A_1 H_2 - h_{\theta_2}\right\| + \delta_2\end{aligned}$$

Further, now we expand $\left\|A_1 H_2 - h_{\theta_2}\right\|$, it follows that

$$\begin{aligned}\left\|A_1(H_2 + H_1 - H_1) - h_{\theta_2}\right\| &\leq \left\|A_1 H_1 - h_{\theta_2} + A_1(R_2 - R_1)\right\|, \\ &\leq \delta_1 + \Delta + \|A_1\|_{\text{op}} d_{\text{repr}}(R_1, R_2^\star).\end{aligned}$$

By the same derivation for the other side, we yield that

$$d_{\text{repr}}(R_2^\star, R_2) \leq \max\left(\|B_2\|_{\text{op}}(\delta_1 + \Delta + \|A_2\|_{\text{op}} d_{\text{repr}}(R_1, R_2^\star)) + \delta_2, \delta_1 + d_{\text{repr}}(R_1, R_2^\star) + \|B_2\|_{\text{op}}(\Delta + \delta_2)\right).$$

Summing both of these upperbounds, we yield the theorem. □

Where is faithfulness in all of this? Unintuitively, neither Main Result 1 nor 2 contain the faithfulness of A_1, A_2 (η_1, η_2, respectively). Our insight is that faithfulness is implicitly encoded into the alignment classes $\Pi, \Pi_\star$. This is because any valid alignment must be consistent with the faithfulness of A_1, A_2 (see Definition A.7). In this way, η_1, η_2 determine which alignment classes are non-empty. This is why alignment classes form the crux of our constructions, because they describe both structural and behavioral constraints over abstractions.

G Computing Interpretive Equivalence

Definition G.1 (Ambiguity)**.** Let $F_1, F_2 \sim \Pi_1^{-1}(A_1)$ and $F_3, F_4 \sim \Pi_2^{-1}(A_2)$ then define

$$p_1 \triangleq \mathbb{P}[d_{\text{repr}}(F_1, F_2) < d_{\text{repr}}(F_1, F_3)] \qquad p_2 \triangleq \mathbb{P}[d_{\text{repr}}(F_3, F_4) < d_{\text{repr}}(F_3, F_1)]. \tag{G.1}$$

Then, the **ambiguity** between $\Pi_1^{-1}(A_1), \Pi_2^{-1}(A_2)$ is then $1 - |p_1 + p_2 - 1|$.

In words, we sample two implementations from $\Pi_1^{-1}(A_1)$ and one from $\Pi_2^{-1}(A_2)$. Then, ambiguity is the probability that the two A_1 representations are more similar to each other than either does to the A_2 implementation. If $\Pi_1^{-1}(A_1) = \Pi_2^{-1}(A_2)$, by symmetry, $p_1 = p_2 = 1/2$ and ambiguity is 1: from their representations alone the implementations of A_1, A_2 are maximally ambiguous.

G.1 Proof for Main Result 3

Proof. Let $F_1, F_2 \sim \Pi_1^{-1}(A_1)$ and $F_3 \sim \Pi_2^{-1}(A_2)$, then consider the event $d_{\text{repr}}(F_1, F_2) > d_{\text{repr}}(F_1, F_3)$. For arbitrary $\varepsilon > 0$, this event occurs with probability 1 when $d_{\text{repr}}(F_1, F_2) > \kappa(A_1, K_1, \Pi_1) + \varepsilon$ and $d_{\text{repr}}(F_1, F_3) < d_{\text{interp}}(A_1, A_2) - \varepsilon$. Therefore,

$$p \geq 1 - \mathbb{P}[d_{\text{repr}}(F_1, F_2) > \kappa(A_1, K_1, \Pi_1) + \varepsilon] - \mathbb{P}[d_{\text{repr}}(F_1, F_3) < d_{\text{interp}}(A_1, A_2) - \varepsilon],$$

since $\varepsilon > 0$ was arbitrary, we yield the result. □

G.2 Intervention-Implementation Duality

Our results in the previous section demonstrate that estimating both the representation similarity and interpretive compression is crucial to understanding interpretive equivalence. If $\|\cdot\| = \|\cdot\|_2$ which is what we will use in the following sections, existing theory provide strong guarantees on how quickly and accurately we can estimate $d_{\text{repr}}(R_1, R_2)$, a multi-target linear regression. However, it is unclear how interpretive compression, essentially a worst case condition can be estimated tractably. We now show that by viewing each intervention as instantiating a new implementation in the space of implementations we only need $\log(n)$ number of interventions where n is the covering number of the implementation space.

Proposition G.1. *Suppose that $C = N(\Pi^{-1}(A), d, \varepsilon/2)$ be the covering number of the implementation space $\Pi^{-1}(A)$. Let $B_1, \ldots, B_C$ be the balls of radius $\varepsilon/2$ that cover $\Pi^{-1}(A)$. Define*

$$p_{\min} \triangleq \min_{1,\ldots,C} \mathbb{P}[B_i],$$

Then, for

$$m \geq \frac{\log(C) - \ln(\delta)}{p_{\min}},$$

we have that $\mathbb{P}[\kappa(A,K,\Pi) - \hat{\kappa}(A,K,\Pi) \leq \varepsilon] > 1 - \delta$ where $\hat{\kappa}$ is the empirical diameter of $\Pi^{-1}(A)$.

Proof. Consider the bad event $\kappa(A,K,\Pi) - \hat{\kappa}(A,K,\Pi) > \varepsilon$. This occurs when we draw m samples such that there exists one ball B_i where we did not draw a sample from. Notice that this occurs with probability at most $(1 - p_{\min})^m$. By the Poisson approximation, we have that $(1 - p_{\min})^m \leq e^{-m p_{\min}}$. Setting this less than δ and rearranging, we yield our desired bound. □