Circuit Stability Characterizes Language Model Generalization

Alan Sun
Carnegie Mellon University
alansun@andrew.cmu.edu

Abstract

Extensively evaluating the capabilities of (large) language models is difficult. Rapid development of state-of-the-art models induce benchmark saturation, while creating more challenging datasets is labor-intensive. Inspired by the recent developments in mechanistic interpretability, we introduce *circuit stability* as a new way to assess model performance. Circuit stability refers to a model's ability to apply a consistent reasoning process—its circuit—across various inputs. We mathematically formalize circuit stability and *circuit equivalence*. Then, through three case studies, we empirically show that circuit stability and the lack thereof can characterize and predict different aspects of generalization. Our proposed methods offer a step towards rigorously relating the generality of models to their interpretability[1].

1 Introduction

Though there exists a wealth of theoretical techniques to analyze and predict generalization, empirically benchmarking a given language model's capabilities remains difficult. This gap between theory and practice stems, in part, from the rapid saturation of existing benchmarks and also the labor-intensive process of creating new, more challenging datasets (Srivastava et al., 2023; Jimenez et al., 2024; Glazer et al., 2024). One way to sidestep these issues is to evaluate a specific capability shared among many tasks. For example, needle-in-the-haystack evaluates language models' ability to perform long-context recall (Kamradt, 2023), while SKILL-MIX measures skill composition performance (Arora and Goyal, 2023; Yu et al., 2024). Although such datasets can be automatically generated and scaled in difficulty, a fundamental challenge remains: identifying specific capabilities that are meaningful to benchmark in the first place. Even after identifying a capability-of-interest, creating salient tasks that precisely target this capability is nontrivial.

In this paper, we seek to address some of these issues by introducing the concept of *circuit stability*. Informally, for a fixed task and model, circuit stability is the consistency of a model's reasoning process (its circuit) across collections of subtasks. Consider solving an algorithmic problem like parity. We expect a strong model to learn and correctly apply a consistent algorithm regardless of the length of its input. If, however, the model learns a different algorithm for each input length, its finite capacity will guarantee a length past which the model will fail.

At the core of our approach are three key insights. First, orthogonal to previous approaches that evaluate a model's performance on small, finite sets of examples by testing inputs one-by-one, we extract and analyze the model's circuit using techniques from mechanistic interpretability (Olah et al., 2020). Since the extracted circuit can be reused and applied by the model to an infinite class of examples, its stability could be a more robust estimator of performance. This is analogous to the formal verification of an algorithm where we do not need to verify every input-output pair (Cousot and Cousot, 1977). Additionally, instead of specifying individual, potentially contentious skills-of-interest, our approach makes a simplifying assumption that a learned skill/circuit is useful only if it is consistently applied by the model. Finally, unlike mechanistic interpretability approaches that seek to intractably extract *hard circuits*, which are discrete subsets of the model's computational graph, we introduce a continuous relaxation: *soft circuits*. This preserves rich structural insights while enabling more easy computation.

Concretely, our contributions are four-fold:

- We formally define circuit stability (Section 3).
- Through two case studies on arithmetic reason-

[1]The codebase for our experiments can be found at https://github.com/alansun17904/circuit-stability

ing and Boolean expression evaluation, we show that circuit stability can predict length, structural, and compositional generalization (Section 4 and Section 5, respectively).

- We show that circuit stability also predicts generalization even on tasks that are not naturally algorithmic like sports understanding (Suzgun et al., 2023).
- We demonstrate that circuit stability can be induced through prompting methods like chain-of-thought (Section 6).

In this paper, we focus our analyses of circuit stability on language models based on the Transformer architecture (Vaswani et al., 2017). Nevertheless, the formal framework we introduce is modality- and architecture-independent. We discuss this further in Appendix A.

2 Background

Herein, we briefly review the circuits framework and relevant concepts in mechanistic interpretability, as they lay the foundation for our contribution.

Circuits. The goal of the circuits framework is to interpret the decision processes of a neural network. This is typically done through two processes: first, identify a minimal subset of the model's computational graph that is responsible for a specific behavior; second, assign human-interpretable explanations to each of the extracted components. The former is referred to as *circuit discovery* while the latter is called *mechanistic interpretability*.

We represent a Transformer's computational graph using the framework introduced by (Elhage et al., 2021; Conmy et al., 2023). Specifically, we view each MLP layer as single node and, unless otherwise specified, we also split each attention head into four distinct nodes: key, query, value, and output. A directed edge is drawn from nodes $n_i \longrightarrow n_j$ if the output of n_i is directly used as an input to n_j. A circuit is defined as a computational subgraph.

Finding Subcircuits. Let $G^M = (V^M, E^M)$ be the computational graph for a model M. For a fixed task and model performance metric L, circuit discovery methods search for a *minimal* computational subgraph: $g^M = (v^M, e^M)$, where $v^M \subset V^M, e^M \subset E^M$. Informally, after ablating the edges and nodes not in g^M, g^M must maintain model performance within a specified margin of $\varepsilon > 0$ (Wang et al., 2022; Shi et al., 2024). Alternatively, one can also view this process as searching for a binary function, $c : E^M \longrightarrow \{0, 1\}$, subject to the aforementioned constraints. We refer to c as a *hard circuit* because any given edge in the computational graph takes on a binary state. Searching for c is known to be an intractable combinatorial optimization problem (Adolfi et al., 2025). As a result, many approximations to circuit discovery have been derived (Nanda, 2023; Hanna et al., 2024; Bhaskar et al., 2024). Even then, it has been shown that the discovered circuit may be highly sensitive to L and ε, the performance metric and threshold, respectively (Conmy et al., 2023; Miller et al., 2024). In this paper, we partially circumvent these issues by redefining a circuit as a mapping $c : E^M \longrightarrow \mathbb{R}$. For each $e \in E^M$, $c(e)$ represents the change in L after ablating e from E^M, essentially capturing the importance of e. In this way, we yield a *soft circuit*. We formalize this in Section 3.

3 Circuit Stability and Equivalence

In this section, we formally define circuit stability. First, we define the notion of a task (Definition 1). Then, tasks are equipped with variable substructure through subtasks (Definition 2). Building on these subtasks, we define the three key concepts of this paper: soft circuitry (Definition 3), ε-circuit stability (Definition 4) and α-circuit equivalence (Definition 5).

Let $\mathcal{X}, \mathcal{Y}$ be the space of all finite input and output strings (Du et al., 2023; Cotterell et al., 2023). The exact construction of these spaces are unimportant. We only require a distribution over $\mathcal{X} \times \mathcal{Y}$ which we call a task.

Definition 1 (Task)**.** *A **task** is a distribution over $\mathcal{X} \times \mathcal{Y}$ denoted by $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$. This is also called the **data distribution**.*

A task may itself contain rich substructure. For even a simple task such as two-operand addition[2], we can, naturally decompose this into many distinct collections of subtasks by simply partitioning the input-output space. One way is to separate addition problems that require carrying at least one digit versus ones that do not. On the other hand, we could also create another collection of subtasks by varying the number of digits in each operand. Intuitively, an appropriate partitioning of the input-output space should yield a collection of subtasks

[2]In this case, the marginal of $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ over $\mathcal{X}$ assigns positive measure to a subset $X \subset \mathcal{X}$ only if all $x \in X$ follows the form "`a + b = `" where $a, b \in \mathbb{Z}^{\geq 0}$. And, the conditional distribution of $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ on $\mathcal{Y}$ given "`a + b = `" is a point mass on the string $a + b$.

that make clear the necessary capabilities a model must have to solve the task itself. For example, success over the aforementioned partitions could indicate both compositional and length generalization, respectively (Wiedemer et al., 2023). Without the loss of generality, we assume that all cells of any subsequently discussed partition have positive measure under $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$. In this way, each cell contains meaningful examples and skills that a model must master in order to achieve perfect performance. By way of its cells, a given partition also elicits its own set of tasks. We call these conditional distributions *subtasks*.

Definition 2 (Subtask)**.** *For a task, $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ and partition $\mathcal{S}$ of $\mathcal{X}\times\mathcal{Y}$. A subtask, over a cell $s \in \mathcal{S}$, is the conditional distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}|s$. For brevity, we notate this as $\mathcal{D}_s$.*

Through $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, we can also measure the importance of any subtask by leveraging the marginal distribution over $\mathcal{S}$. We call this distribution the *partition distribution* and notate it as $\mathbb{P}_{\mathcal{S}}$.

We now precisely define a model's *soft circuit* relative to a task (or subtask). Our approach differs from the traditional notion of a circuit defined in mechanistic interpretability. Rather than assign a binary indicator to the edges of the computational graph, $E^M \longrightarrow \{0,1\}$, we perform a continuous relaxation. A comparative analysis of this setting, along with its implications, are provided in Section 8.

Definition 3 (Soft Circuit)**.** *Consider a task $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, a model $M : \mathcal{X} \longrightarrow \mathcal{Y}$ with computational graph $G^M = (V^M, E^M)$, and some performance metric $L : \mathcal{Y}\times\mathcal{Y} \longrightarrow \mathbb{R}$. With respect to $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, M's **soft circuit** is a function $c : E^M \longrightarrow \mathbb{R}$ such that for any $e \in E^M$,*

$$c(e) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}}[L(M_{\{e\}}(x), y) - L(M(x), y)], \quad (1)$$

where $M_{\{e\}}$ denotes M after ablating edge e.

As long as L is well-defined, c always exists. We defer the technical details of L and the ablation procedure to Appendix B. By our previous constructions, a subtask (Definition 2) may also induce a soft circuit. So for any subtask $\mathcal{D}_s$, we denote its induced soft circuit as c_s.

Intuitively, $c(e)$ captures the singular importance of e. By examining and comparing the collective mapping, c, across subtasks, we gain insight into holistic model behavior. Therefore, we take $K : \mathbb{R}^{E^M} \times \mathbb{R}^{E^M} \longrightarrow \mathbb{R}$ to be a measure of the similarity between two soft circuits: a kernel-like function[3]. We are now ready to define circuit stability.

Definition 4 (ε-Circuit Stable)**.** *For $\varepsilon > 0$, a model, M, is **ε-circuit stable** with respect to a task $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ and a collection of partitions $\mathscr{P}$ if*

$$\inf_{\mathcal{S}\in\mathscr{P}} \mathbb{E}_{s,s'\sim\mathcal{S}}[K(c_s, c_{s'})] > \varepsilon, \quad (2)$$

where s, s' are two subtasks sampled i.i.d. from the partition distribution $\mathbb{P}_{\mathcal{S}}$.

In Equation 2, the expression inside the infimum measures the stability of a model's soft circuit as we move between different subtasks. This stability is weighted by the partition distribution. Thus, if two subtasks have a low probability of occurring with respect to $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, we consider the skills required to solve them unimportant. In turn, instability across these subtasks is also disregarded. This allows us to avoid specifying *a priori* which skills are important for analysis.

For all our experiments, we take K to be Spearman's ρ. Concretely, we take soft circuits $c_s, c_{s'}$ and individually induce a ranking of E^M through $c_s(E^M), c_{s'}(E^M)$. Then, we measure the correlation coefficient between these ranks. Throughout, we construct collections of partitions $\mathscr{P}$ manually, based on the task at hand. In Section 6 and 8, we discuss constructing partitions statistically.

Next, using K we also define a type of pointwise-equivalence between the soft circuits of any two subtasks.

Definition 5 (α-Equivalent)**.** *For $\alpha > 0$, soft circuits $c_s, c_{s'}$ are **α-equivalent** if $K(c_s, c'_s) \geq \alpha$.*

4 Case Study: Arithmetic Reasoning

In this section, we explore the circuit stability and equivalence of `gemma-2-2b`[4] over the task of two operand addition (Rivière et al., 2024). These problems come in the form of "`a + b =` ", where $a, b \in \mathbb{Z}^{\geq 0}$. We examine a specific partition where each subtask contains problems where the values of a all have the same number of digits, and similarly for b. This setup allows us to study two different

[3] K should be thought of as a reproducing kernel Hilbert space kernel over the function space $\mathbb{R}^{E_M}$. However, for simplicity, our experiments do not adhere to this guiding principle. Instead we use a more interpretable similarity metric such as rank correlation. A deeper investigation into the theoretical properties of K and its implications for circuit stability is left for future work

[4] 2 billion parameter model containing over 79k circuit edges as defined in Section 2.

forms of generalization that have been separately analyzed in the literature: length generalization, where the number of digits in a, b increase independently (Cho et al., 2025), and compositional generalization which tests whether models solve addition problems recursively (Kudo et al., 2023; Nikankin et al., 2025). We find that `gemma-2-2b`'s circuit instability across subtasks correlates with fluctuations in its performance on those same subtasks, indicating a potential causal link between circuit instability and generalization failures.

4.1 Experimental Setup

We denote a subtask as an ordered pair (o_1, o_2) where $1 \leq o_1, o_2 \leq 8$. o_1, o_2 denote the number of digits in a, b respectively. Over all experimental settings, we provide the model with $k = 3$ few-shot examples. To implement circuit discovery (Definition 3), we choose L to be the next-token patching metric, defined in Equation 8. Each edge is ablated through noisy-to-clean patching using both noisy and clean samples from the same subtask. These design choices are well-documented in Heimersheim and Nanda (2024) and we explore their implications in Appendix B. We perform circuit discovery over each subtask, resulting in 64 soft circuits for analysis.

4.2 Identifying Arithmetic Circuit Families

To analyze the relationships between subtasks, we compute Spearman's ρ, K, between their soft circuits. Using $\alpha = 0.6$, we apply the notion of α-equivalence, as defined in Definition 5, to cluster the arithmetic subtasks into roughly five distinct clusters: equal-digit ($o_1 = o_2$), one-digit difference ($o_1 = o_2 \pm 1$), leading-operand heavy ($o_1 > o_2$), single-digit ($o_2 = 1$), and trailing-operand heavy ($o_2 > o_1$). These subtask clusters are visualized in Figure 1(**top**).

To confirm that these clusters are not merely an artifact of a particular setting of α, we perform two complementary experiments. First, we directly compute a set of t-SNE embeddings using all the soft circuits (Maaten and Hinton, 2008). Notably, these embeddings are independent of α. The relative distances between the embedded circuits are visualized in Figure 1(**bot**). We find that the circuit clusters we identified before form well-separated groups under this representation as well.

Next, since we take K to be Spearman's ρ, α is naturally bounded between $[-1, 1]$. Consequently, we expect the number of distinct subtask clusters to

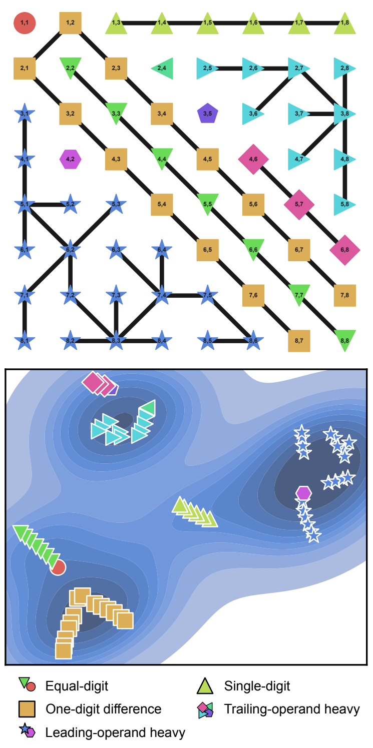

Figure 1: (**top**) Each node represents a distinct subtask. An undirected path exists between two nodes if and only if they are α-equivalence for $\alpha = 0.6$. (**bot**) t-SNE embeddings of the soft circuits for each subtask with `perplexity=3`. The node shape and color combinations are consistent with the circuit clusters in (top).

increase monotonically as α also increases monotonically. This behavior is shown in Figure 2. As α approaches its upper bound of 1, the number of subtask clusters converges to the total number of subtasks. This indicates that no two subtasks have identical circuits. On the other hand, for $\alpha = 0.4$, only one subtask cluster exists. This suggests the presence of a common set of circuit components that are important for arithmetic generally. In other words, no two distinct subtasks reply on entirely disjoint sets of components. This observation aligns with the previous findings of Stolfo et al. (2023); Hanna et al. (2023); Nikankin et al. (2025) that the numerical abilities of language models are mediated by a common set of attention heads and MLPs.

Surprisingly, there exists a critical threshold $\alpha = 0.6$ where the number of circuit families explodes (see the red region in Figure 2). We visualize the circuit families in this critical region in Figure 3. As α increases from 0.5 to 0.53, a clear separation immediately forms between the trailing-operand and leading-operand heavy subtasks. This could indicate fundamental differences in how the model is handling inputs from these two subtasks. As α continues to increase, we observe the emer-

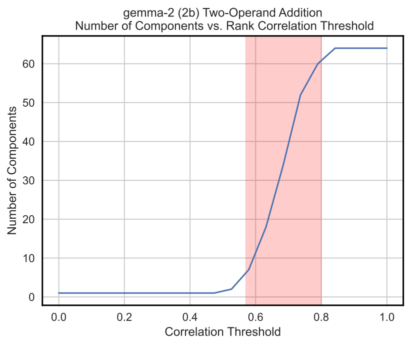

Figure 2: The number of α-circuit equivalence families as α varies between $[0, 1]$. We omit visualization of $\alpha < 0$ since the number of circuit families is a monotonic function of α. The red region shows that 80% of circuit families emerge between $\alpha = 0.58$ and $\alpha = 0.79$

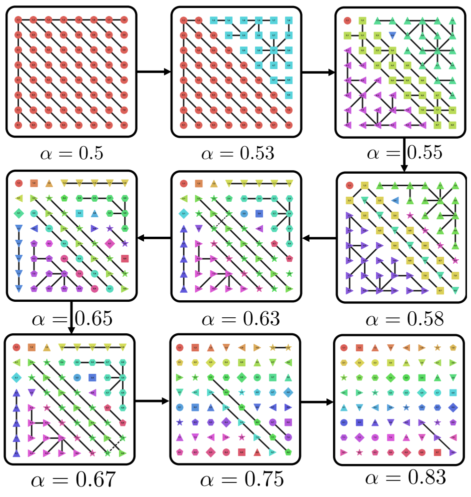

Figure 3: The emergent families of α-equivalent subtasks as α varies between $[0, 1]$.

gence and persistence of distinct families corresponding to equal-digit, single-digit, and one-digit difference subtasks. The stability of these families over increasing α suggests strong internal cohesion, further validating our proposed clustering depicted in Figure 1**(bot)**.

4.3 Circuit Stability and Generalization

The family of α-equivalences (Definition 5) identified in the previous subsection deviate from our expected clustering of a well-performing model (Section 3). In particular, a model that truly understands two-operand addition should necessarily recognize that addition is both commutative and associative. In this section, we argue that lack of α-equivalence across these aforementioned circuit families indicates that the model neither adheres to nor fully internalizes these properties of addition.

First, an understanding and application of commutativity implies that any subtask (o_1, o_2) should be equivalent both in performance and circuitry to the subtask (o_2, o_1). This is because a model that learns this axiom could accordingly transpose the two operands before adding, thereby achieving consistent performance across this collections of subtasks. However, as discussed previously, a lack of α-equivalence between leading and trailing-digit heavy families suggest that `gemma-2-2b` does not respect commutativity. As a result, we expect a significant performance gap as we move between these subtask families. Indeed we observe this to be the case. We benchmark `gemma-2-2b` across all 64 subtasks. Per task, $n = 1000$ problems are sampled independently while maintaining the same formatting scheme as before (see Section 4.1). Performance is measured through exact string match accuracy (Srivastava et al., 2023). We find that `gemma-2-2b`'s performance positively skews towards leading-digit heavy subtasks (see Figure 4). In some cases, the performance difference between (o_1, o_2) and (o_2, o_1) can be more than 20%.

Next, the associativity of addition implies that any two-operand addition problem can be decomposed into a sequence of $(1, 1)$ problems. More generally, a subtask like $(8, 2)$ could also be broken down into a sequence of $(8, 1)$ and $(1, 1)$ problems. Likewise, $(6, 7)$ can be decomposed into $(6, 6)$ and $(1, 1)$. If the model is leveraging associativity, we would expect its errors to also compound in a predictable manner due to the repeated reuse of simpler subtasks. But, we observe through the previous subsection, that even adjacent subtasks like $(8, 1)$ vs. $(8, 2)$ or $(6, 6)$ vs. $(6, 7)$ belong to different α-equivalent clusters. This suggests that `gemma-2-2b` is not exploiting associativity to systemically reuse its circuit components across subtasks. This behavior is verified quantitatively in Figure 4, where model performance steeply drops off across subtasks $(6, 6), (6, 7)$, etc.

Lastly, we hypothesize that within an α-equivalent cluster, `gemma-2-2b` *is* reusing its circuit components. This behavior has been previously identified in other tasks (Merullo et al., 2023). We find that hard circuits within the same α-equivalent cluster share a large number of components or, in some cases, even function as subcircuits of one another (see an example in Figure 5). Here, we greedily construct[5] hard circuits by assigning the top 200 components—as given by the soft circuitry—

[5]Though this is a naive decoding method, empirically it works quite well as an approximation for the actual circuit (Conmy et al., 2023; Hanna et al., 2024).

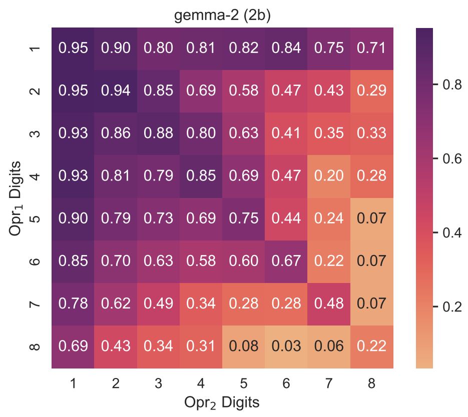

Figure 4: Performance of gemma-2-2b over arithmetic subtasks as Opr$_1$:= o_1 and Opr$_2$:= o_2 increase. Each cell denotes the exact string match accuracy.

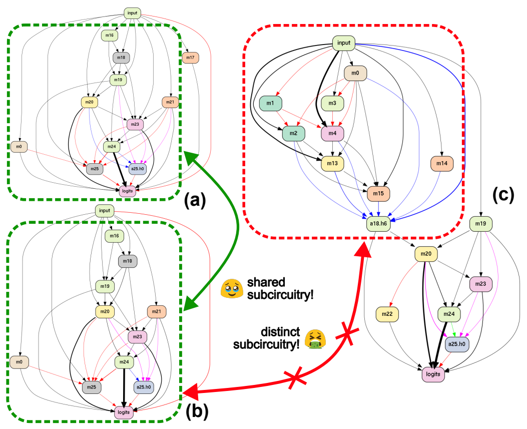

Figure 5: Circuits for **(a)** $(8,8)$ **(b)** $(7,7)$ and **(c)** $(2,7)$. **(a)** and **(b)** share many subcircuit components and are α-equivalent for $\alpha = 0.6$. On the other hand, (c) is not α-circuit equivalent with either (a) or (b).

one (in circuit) and the remaining zero (out of circuit). In contrast to the sharp performance change across non-equivalent subtasks, we find that within α-equivalent subtasks model performance decays smoothly. This degradation in performance can be characterized using tight-fitting regressions that depend on the number of subtask compositions and their associated error rates. We provide a detailed analysis of this in Appendix C.

By combining our insights derived from circuit stability analysis with the benchmark results in Figure 4, we can affirm that the measured performance differences between subtasks are not merely an artifact of statistical noise.

Circuit stability and the lack thereof also point to tangible ways that we can improve the model. For example, during training, circuit stability could possibly be improved through causal alignment (Geiger et al., 2024; Gupta et al., 2024). Alternatively, stability could also be induced at inference via prompting. By explicitly breaking down a complex problem into simpler ones, we could encourage component/subtask reuse. We explore this latter possibility in Section 6 through chain-of-thought prompting.

5 Case Study: Boolean Expressions

We now extend our analysis of circuit stability to a different task that also exhibits rich subtask structure: Boolean expression evaluation (Suzgun et al., 2023). Previously, we argued that circuit stability implies both length and compositional generalization. Here, we refine this perspective by showing that circuit non-equivalence or instability can also provide meaingful insights. Specifically, deviations in stability may indicate structural generalization (Ye et al., 2021; He et al., 2024). In other words, circuit stability is not simply a matter of "more is better;" rather, its desirability depends on how well it aligns with our prior knowledge of the task.

5.1 Experimental Setup

We use `phi-1.5`[6] due to its strong performance on logical and mathematical reasoning (Li et al., 2023). We follow the same evaluation setup as Srivastava et al. (2023) for Boolean expression evaluation and prompt the model with $k = 3$ few-shot examples. We construct partitions based on three independent variables: (1) expression length (number of words, e.g., `True and False` has length 3); (2) parenthetical depth (number of maximum nested parentheses, e.g., `(not (True))` has depth 2); and (3) the set of logical operators used (`not, and, or`). Expression length ranges from 1 to 9, and depth from 0 to 6. Circuit discovery details are largely the same as Section 4.1 and can be found in Appendix B and D.

5.2 Circuit Instability and Generalization

Not Subtask. Consider a Boolean expression that contains only the literals `True`, `False`, and the operator `not`. Since `not` is associative, adding and removing an arbitrary number of parentheses to any expression of this form should not change its ground-truth label. Thus, we expect a model that understands this axiom to apply the same circuit whether or not there are parentheticals in the expression. To test if this holds for `phi-1.5`, we first benchmark its circuit stability separately for sets of expressions with and without parentheses.

[6]A 1.5 billion parameter model with 128k circuit components.

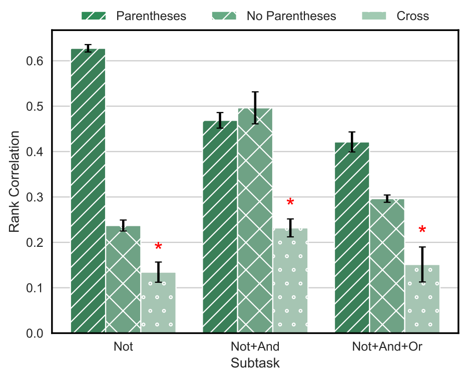

Figure 6: Circuit stability for `phi-1.5` within and across six subtasks. A permutation test is performed between parenthetical and non-parenthetical pairs of subtasks. "*" denotes a statistically significant difference with a setting of $p < 0.05$. The error bars denote a 95% confidence interval.

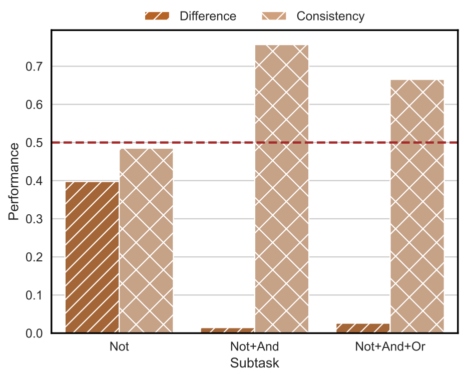

Figure 7: **(left columns)** The performance difference after adding parentheses. **(right columns)** The self-consistency of `phi-1.5` as a result of adding parentheticals. The dotted line denotes random chance of 0.5 for the consistency estimates.

Concretely, for `not` expressions *with* parentheses, we partition the input space by both parenthetical depth and expression length. In contrast, for `not` expressions *without* parentheses, we partition only by expression length. The left two columns of the first bar group in Figure 6 illustrate `phi-1.5`'s respective circuit stability across these two partitions.

The separately evaluated circuit stability of `phi-1.5` on each task provides a baseline for computing α-equivalence. If `phi-1.5` applies the same set of circuits across both parenthetical and non-parenthetical subtasks, then circuit stability should be consistent even as we permute the subtask soft circuits between these two groups. In particular, we expect this permutation not to cause circuit stability to drop below the minimum stability observed in across the two partitions. We find that `phi-1.5` applies statistically significant different circuits between these two tasks (see rightmost bar of first bar group in Figure 6).

We hypothesize that this lack of circuit equivalence suggests that the model does not understand associativity in `not` evaluation. Indeed this is the case. Given any expression containing only `nots` and literals, after adding parentheses, `phi-1.5`'s performance decreases by 40%. Further, the model is not self-consistent: adding parentheses to any expression causes the model to flip its prediction. This behavior is illustrated in the first bar group of Figure 7.

Not+And Subtask. Now consider a Boolean expression with logical operators `not` and `and`. Adding and removing parentheses from this expression changes the order of evaluation. This may flip the ground-truth label[7]. Consequently, a model that respects operator precedence *should* apply substantially different circuits across `not+and` expressions with and without parentheses. We apply the same experimental procedure as the previous `not` subtask. As shown in the second bar group of Figure 6, we find that `phi-1.5` does employ different soft circuits across these partitions. Accordingly, in Figure 7, we observe that `phi-1.5`'s performance stabilizes between subtasks with and without parentheses. Additionally, `phi-1.5` exhibits increased self-consistency. That is, for any particular expression, adding parentheses does not change the correctness of its prediction.

Not+And+Or Subtask. Similar to the previous subtask, adding parentheses to an expression containing the operators `not`, `and`, and `or` alters the order of evaluation. As before, we observe that `phi-1.5`'s circuits align with our expectations (see third bar group of Figure 6). As a result, we see consistent performance stability and increased self-consistency (see rightmost bar group in Figure 7).

6 Case Study: Chain-of-Thought

In Section 4 and 5, we demonstrated that the stability of a model's circuit sheds light on its generalization. Now, we examine methods that tractably induce circuit stability. We hypothesize that chain-of-thought improves performance by promoting subtask decomposition and circuit component reuse (Wei et al., 2022). As a result, we expect chain-of-thought to substantially improve circuit stability. Herein, we present some preliminary evidence for this claim. Unlike previous

[7] `(not False) and True != not False and True`

sections, we examine a task that is knowledge-based: sports understanding (Suzgun et al., 2023). Sports understanding is a binary classification task which presents models with sports statements and the model needs to decided whether they are true or false[8]. We employ both `Llama-3.1-8b` and `Gemma-2-9b`[9] for this case study. Both models are sufficiently large to show significant performance improvements after prompting with chain-of-thought, see Figure 8**(left)**. As before, model performance is also measured using exact string match accuracy.

In contrast to our previous case studies, the subtask structure of this task much less apparent. As a result, we opt to construct subtasks simply by randomly partitioning the dataset into five disjoint cells. We compute the average circuit stability pairwise across these five cells before (we use few-shot prompting with $k = 3$) and after chain-of-thought prompting. These results are shown in Figure 8**(right)**. Across both models, we see that chain-of-thought significantly circuit stability.

It should be noted that technically the partition strategy we employ herein is not creating true subtasks (see Definition 2). This is because we are sampling from $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ a finite dataset *first*, *then* randomly partitioning the resulting dataset. As a result, the partitions we yield are i.i.d. with respect to $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$. Thus, in some sense we are measuring the variance of the soft circuitry distribution before and after applying chain-of-thought. We could remedy this by fixing a partition strategy *a priori* that with high probability induces similar substasks (in terms of transport distance). For example, we could partition subtasks based on the value of the fifth character of the input prompt. The connections between these two approaches could be explored more in future work.

7 Related Works

Most current work in mechanistic interpretability relies on *ad hoc* interpretations tailored for a fixed task and model (Wang et al., 2022; Stolfo et al., 2023; Conmy et al., 2023; Hanna et al., 2023; Arditi et al., 2024; Lee et al., 2024; Nikankin et al., 2025, *inter alia*). As a result, generalizing these results into actionable insights remains difficult. To address this limitation, recent research has studied the dynamics of circuits and their mechanisms. This is also the focus of our paper. For example, works like Nanda et al. (2022); Zhong et al. (2023); Humayun et al. (2024); He et al. (2024); Tigges et al. (2024) examine circuits across the training horizon which shed light on circuit formation and phenomena such as grokking (Power et al., 2022). Further studies such as Lee et al. (2024) seek to compare the tangible mechanistic differences before and after applying post-training methods like alignment. On the other hand, works such as Lieberum et al. (2023); Wu et al. (2023); Merullo et al. (2023) examine the change in a model's circuit with respect to scaling. Perhaps most similar to our line of work is the mechanistic interpretability of skill composition (Arora and Goyal, 2023; Yu et al., 2024) which investigates how models combine learned skills to solve novel problems. Notably, Chughtai et al. (2023); He et al. (2024) analyzes the emergence of skill composition in modular addition. However, they focus on small, toy models. It is unclear how their arguments and conclusions generalize to pretrained language models. In contrast, our framework of circuit stability provides a general characterization of circuits, their equivalence, and their stability—independent of the task or any specific model.

8 Conclusion and Discussion

In this paper, we introduce and formally define *circuit stability* and *equivalence* (Section 3). We provide empirical evidence that circuit stability characterizes many key aspects of generalization and argue that this type of stability is actionable (Section 4 and 5). For example, it can be induced

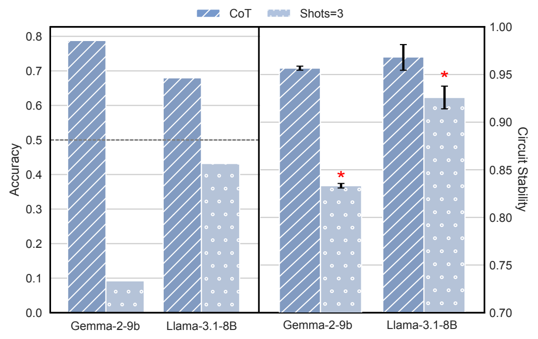

Figure 8: **(left)** Exact string match accuracy of sports understanding task under chain-of-thought versus few-shot prompting. **(right)** Circuit stability across random partitions. We perform a two-sample t-test, "$*$" denotes a significant difference ($p < 0.05$).

[8]For example, the model is presented with a statement like "Santi Cazorla scored a touchdown." This statement is *false* because Santi Cazorla is a soccer player and a "touchdown" is a part of American football and rugby (Suzgun et al., 2023).

[9]8 billion parameter model containing over 1.5m circuit edges and 9 billion parameter model with over 720k circuit edges, respectively.

through prompting (Section 6). Our approach, based on methods from mechanistic interpretability, offers a step towards rigorously bridging the generality of models with their interpretability.

Efficiency. Our definition of a soft circuit (Definition 3) involves continuous relaxation of the binary circuit function, $c : E_M \to \{0, 1\}$. In most circuit discovery procedures, this relaxation, implemented via Equation 3, is necessary for tractability (Conmy et al., 2023; Nanda, 2023; Syed et al., 2024). As a result, many existing methods first perform this relaxation, then apply a greedy decoding strategy to extract the binary circuit function. Thus, our choice of both this continuous relaxation and Spearman's ρ aligns with the standard practices of circuit discovery. Approximations for Definition 3 are also computationally efficient. Methods such as Syed et al. (2024); Hanna et al. (2024) only require (a constant factor of) two forward and one backward pass of the model to estimate the entire soft circuitry.

Occam's Razor. Crucially, soft circuitry does not account for higher level algorithmic similarities. That is, we could have different soft circuits which also correspond to distinct hard circuits, but algorithmically they implement the same procedure (Olsson et al., 2022; Merullo et al., 2023). We do not see this as a limitation of the work but rather a feature of our mathematical framework. From a learning-theoretic perspective, duplicate mechanisms necessarily imply longer minimum description lengths. In turn, this leads to looser generalization bounds (Hansen and and, 2001; Sefidgaran et al., 2023, *inter alia*). More informally, if we subscribe to the principle of Occam's Razor (Blumer et al., 1987; Shalev-Shwartz and Ben-David, 2014) then we would also prefer a model with less duplicate mechanisms. As a result, our analyses implicitly take into account these learning-theoretic constructs. Lastly, if one was truly concerned with algorithmic differences, then K could be augmented using metrics from causal abstraction (Beckers and Halpern, 2019; Geiger et al., 2021; Otsuka and Saigo, 2022; Geiger et al., 2024, *inter alia*). However, this introduces additional complexities that we leave for future work.

Finding Partitions. Throughout Sections 4, 5, we leverage prior knowledge about the task to construct partitions of interest. But, for more complex tasks requiring an intricate composition of skills, the appropriate partitions may not be obvious. In Section 6, we demonstrate that this does not hinder the practicality of circuit stability. Even randomly chosen partitions can yield meaningful insights. Alternatively, since soft circuitry is an expectation, we could have also sought to characterize circuit stability through the asymptotic variance of its limiting distribution. Further, we hypothesize that if any of the partitions contain cells that are ε-representative (Shalev-Shwartz and Ben-David, 2014) with respect to both L and $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$ in Equation 3, then they should lead to a sharp characterization of the model's performance. Investigating this connection could form the basis of interesting future work and reduce reliance on constructing partitions manually.

9 Limitations

Our empirical case studies in Sections 4, 5, and 6 are fairly limited in terms of the tasks and models we benchmark. We hope that these preliminary results will give the larger research community a taste of how circuit stability can be used and leave these extensions for future work.

Another limitation of the work is the choice of circuit abstraction (Vilas et al., 2024). In Section 2, we loosely defined the model's circuit as a subgraph of its computational graph. However, there are many ways this computational graph could be specified. On one extreme, there exists the trivial computation graph: an input and output node with a single edge representing the entire function. On the other extreme, we could define the graph as a trace of the compiled machine code. This is also a challenge that mechanistic interpretability faces. It is unclear how circuit stability would react to these different levels of abstraction. Perhaps future theoretical analyses of circuit stability could take this into account by involving $|V^M|$ into its bounds.

Acknowledgments

Alan thanks Mariya Toneva for her generous support; Fengwen Sun for his valuable feedback and advice; Ethan Sun and Warren Shepard for their contributions to early iterations of the work; and, Andrew Koulogeorge for the thoughtful discussions.

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*.

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. In *Advances in Neural Information Processing Systems*, volume 37, pages 136037–136083. Curran Associates, Inc.

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

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.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding Transformer Circuits With Edge Pruning. In *Advances in Neural Information Processing Systems*, volume 37, pages 18506–18534. Curran Associates, Inc.

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

Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, et al. 2024. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247*.

Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. 2025. Arithmetic transformers can length-generalize in both operand length and count. In *The Thirteenth International Conference on Learning Representations*.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. 2023. A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations. In *Proceedings of the 40th International Conference on Machine Learning*.

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*.

Ryan Cotterell, Anej Svete, Clara Meister, Tianyu Liu, and Li Du. 2023. Formal aspects of language modeling. *arXiv preprint arXiv:2311.04329*.

Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '77, page 238–252, New York, NY, USA. Association for Computing Machinery.

Li Du, Lucas Torroba Hennigen, Tiago Pimentel, Clara Meister, Jason Eisner, and Ryan Cotterell. 2023. A measure-theoretic characterization of tight language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9744–9770, Toronto, Canada. Association for Computational Linguistics.

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.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586. Curran Associates, Inc.

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.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems*, volume 34, pages 572–585. Curran Associates, Inc.

Rohan Gupta, Iván Arcuschin, Thomas Kwa, and Adrià Garriga-Alonso. 2024. InterpBench: Semi-Synthetic Transformers for Evaluating Mechanistic Interpretability Techniques. In *Advances in Neural Information Processing Systems*, volume 37, pages 92922–92951. Curran Associates, Inc.

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*.

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*.

Mark H Hansen and Bin Yu and. 2001. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. 2024. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. 2024. Grokking and the geometry of circuit formation. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Greg Kamradt. 2023. LLM Test - Needle in a Haystack.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. 2024. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*.

Keito Kudo, Yoichi Aoki, Tatsuki Kuribayashi, Ana Brassard, Masashi Yoshikawa, Keisuke Sakaguchi, and Kentaro Inui. 2023. Do deep neural networks capture compositionality in arithmetic reasoning? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1351–1362, Dubrovnik, Croatia. Association for Computational Linguistics.

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*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *Preprint*, arXiv:2309.05463.

Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. 2023. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605.

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*.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2023. Circuit Component Reuse Across Tasks in Transformer Language Models. In *The Twelfth International Conference on Learning Representations*.

Joseph Miller, Bilal Chughtai, and William Saunders. 2024. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling*.

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

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*.

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*.

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

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.

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.

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

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization Beyond Overfitting on Small Datasets.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, et al. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118.

Milad Sefidgaran, Abdellatif Zaidi, and Piotr Krasnowski. 2023. Minimum Description Length and Generalization Guarantees for Representation Learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 1489–1525. Curran Associates, Inc.

Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David M. Blei. 2024. Hypothesis testing the circuit hypothesis in llms. In *Advances in Neural Information Processing Systems*, volume 37, pages 94539–94567. Curran Associates, Inc.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052, Singapore. Association for Computational Linguistics.

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*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US. Association for Computational Linguistics.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

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.

Martina G. Vilas, Federuci Adolfi, David Poeppel, and Gemma Roig. 2024. Position: An Inner Interpretability Framework for AI Inspired by Lessons from Cognitive Science. In *Proceedings of the 41st International Conference on Machine Learning*.

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*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. 2023. Compositional generalization from first principles. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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.

Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. 2021. Towards a theoretical framework of out-of-distribution generalization. In *Advances in Neural Information Processing Systems*, volume 34, pages 23519–23531. Curran Associates, Inc.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. 2024. SKILL-MIX: a flexible and expandable family of evaluations for AI models. In *The Twelfth International Conference on Learning Representations*.

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*.

A Circuit Stability Beyond Transformer Language Models

In this paper, we focused our case studies on Transformer language models, as their circuits are well-studied in the the literature (Elhage et al., 2021;

Olsson et al., 2022; Wang et al., 2022; Hanna et al., 2023). However, the concept of the circuit stability can be extended to any neural network as long as it admits a consistent computational graph. In this section, we present three distinct neural architectures associated with applications outside of the language domain and demonstrate that even in these cases, circuit stability is a well-defined notion. In this way, circuit stability is a modality-independent concept. An interesting line of future work could be to extend this framework to study the circuits and generalization of vision-language models (Bordes et al., 2024).

Fully-Connected Neural Networks. A fully-connected network consists of successive linear layers with activation functions: $M = f_n \circ \ldots \circ f_1$ where $f_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$. Here, d_1, d_{n+1} are the input and output dimensions. For given activation functions $\sigma_i : \mathbb{R} \to \mathbb{R}$, a single fully-connected layer is defined as $f_i(x) = \sigma_i(W_i x)$, where $W_i \in \mathbb{R}^{d_{i+1} \times d_i}$ and σ_i is applied element-wise. There are many ways to construct a valid computational graph upon this architecture. For brevity, we only show the most intuitive construction: a node in the i^{th} layer is simply an entry in x. Specifically, we can decompose the j^{th} entry in layer i as $(\sigma_i(W_i x))_j = \sigma_i\left(\sum_{k=1}^{d_i} W_{jk} x_k\right)$. Note also that each entry in the input should also be assigned a node. Therefore, this node forms edges with all nodes in the previous layer.

Convolutional Neural Networks. We take a similar approach to the decomposition of fully-connected neural networks. For simplicity, we only consider the case of a single 2d-convolution layer, a convolutional network with higher dimensions or more layers can be derived inductively. We directly lift this description of a convolutional layer from (Sun et al., 2024). Let $f : \mathbb{R}^{c \times h \times w} \to \mathbb{R}^{c' \times h' \times w'}$. Suppose that this layer is parameterized by kernels $W_i \in \mathbb{R}^{k \times k}$ for $1 \leq i \leq c'$ and some $k \in \mathbb{N}$ as well as a bias $b \in \mathbb{R}^{c'}$. Then, it follows that

$$f(x)_j = \left(\mathbf{1}_{h' \times w'} b_j + \sum_{i=1}^{c} W_j * x[i, :, :] \right),$$

for $1 \leq j \leq c'$ where $f(x)_j \in \mathbb{R}^{h' \times w'}$ for h', w' being the resulting dimension after convolution with a $k \times k$ kernel. Here, $\mathbf{1}_{h' \times w'} \in \mathbb{R}^{h' \times w'}$ is a one matrix. Then, one convolutional layer needs $c'h'w' + chw$ nodes: one for each input and output coordinate. A directed edge is drawn from all input nodes to each output node[10].

State-Space Language Models. A state-space model processes input tokens sequentially. Each token, x_t is processed in constant time (Gu et al., 2021) via a hidden state s_t such that

$$\begin{aligned} s_{t+1} &= \boldsymbol{A} s_t + \boldsymbol{B} x_t, \\ y_t &= \boldsymbol{C} s_t + \boldsymbol{D} x_t. \end{aligned}$$

Thus, we can the computational at each time-step to essentially be a combination of two fully-connected neural networks. We can specify a well-defined computational graph by unrolling the recursive equations above. We define each x_t for all t as a node and each entry of s_t to also be a node. The, the edges are given by the same schema we used to define the fully-connected layers.

Putting it All Together. These examples demonstrate that as long as we can specify a computational graph, the circuit of a model is well-defined. Therefore, the existence of circuit stability as well as its implications depends largely on the task distribution (Definition 1), its subtask distributions (Definition 2), and the granularity of the computational graph. To this last point, *a priori* it is unclear what the correct level of abstraction one should impose on the nodes and edges. On one end, we could define the nodes as the input/output of each FLOP of compute. However, though this might provide lots of insight, the computation of its circuit (Definition 3) would be intractable. On the other end, we could define the computational graph as a single node with no edges. Though this is computationally more favorable, it yields no insights. The subfield of causal abstraction addresses some of these issues and we refer the reader to Geiger et al. (2021).

B Circuit Discovery Details

For all of our circuit discovery experiments, we perform *attribution patching* (Nanda, 2023). Attribution patching is a linear approximation to *activation patching* also called causal mediation analysis (Heimersheim and Nanda, 2024). First introduced by Vig et al. (2020) and extended upon by Meng et al. (2022); Wang et al. (2022); Conmy et al. (2023), activation patching seeks to determine the

[10]This may not be the tightest graph that we can build in terms of the number of edges we need to construct. However, after performing any ablations, we should expect those edges that are non-tight to contribute a score of 0.

effect of a single neural component on the entire model's output, for some fixed task. Then, by isolating all such components, we have effectively found the subcircuit responsible for the model's behavior on this specific task. More formally, let $L(\cdot)$ be a function that maps a model's output into a scalar, generically this could be some loss function. This is the *patching metric*. For notational brevity, we omit L's dependency on the ground truth label. Let c be an arbitrary neural network component that we wish to patch, and also denote by x an arbitrary input to our model M. Using Pearl's (2009) notion of do-calculus, activation patching can be written as

$$\text{Patch}_c(x, c^\star) := L(M^{\text{do}(c=c^\star)}(x)) - L(M(x)), \quad (3)$$

where $c^\star$ is a *counterfactual* output of activation c that we patch in. In natural language, $\text{Patch}_c(x, c^\star)$ can be expressed as "if we replace only the output of component c with $c^\star$, how will the model now behave?" We encourage the reader to refer to Heimersheim and Nanda (2024) for a detailed introduction to activation patching and instead briefly explain attribution patching, our design choices, as well as our extensions to the multi-token setting.

One drawback of activation patching is its computational cost. Consider a dataset of n data points and a model with k neural components we are interested in patching, activation patching would require $O(nk)$ forward passes. This becomes prohibitively expensive when $k \gg 1$ (Kramár et al., 2024). Thus, using Equation 3 as a jumping off point, attribution patching seeks to make activation patching more efficient. Consider a first-order Taylor series approximation of L around c assuming that $c^\star \approx c'$, where c' is the unpatched activation of c on x. Then, it follows that

$$\text{Patch}_c(x, c^\star) \approx (c' - c^\star)\nabla_c L(x). \quad (4)$$

Crucially, Nanda (2023); Syed et al. (2024) argue that this new metric can be computed in two forward passes and one backward pass for all components. Thus, we only require $O(n)$ forward and backward passes. Throughout the paper, we use a variant of attribution patching introduced by (Hanna et al., 2024) called *edge attribution patching* with *integrated gradients* (EAP-IG). In short, EAP-IG deduces a more accurate approximation to activation patching than vanilla attribution patching in Equation 4. EA-IG operates by computing a path integral from $c' \longrightarrow c^\star$:

$$\Delta_{c',c^*} \int_0^1 \frac{\partial L}{\partial c'} M^{\text{do}(c=\Delta_{c',-\alpha(c^\star - c')})}(x)\, d\alpha \quad (5)$$

$$\approx \Delta_{c',c^*} \frac{1}{m} \sum_{k=1}^{m} \frac{\partial L}{\partial c'} M^{\text{do}(c=\Delta_{c',-k(c^\star - c')/m})}(x), \quad (6)$$

where $\Delta_{c',c} = c' - c$ and m is a hyperparameter representing the number of steps to approximate the integral. Equation 6 can be understood as a Monte-Carlo estimate of the integral in Equation 5. Generally, for such estimates to be accurate it requires $m \gg 1$, potentially on the order of $\sim 10^5$. However, empirically Hanna et al. (2024) finds even $m = 5$ works quite well. These hyperparameters are adopted for all of the circuit experiments.

B.1 Patching Multi-token Tasks

Herein, we detail circuit discovery in the case of multitoken tasks. To the best of our knowledge, almost all of the mechanistic interpretability literature deals with tasks that require only a single token output. So, our methods represent one attempt to generalize these existing approaches. We present two distinct approaches, and briefly discuss their interpretations (Table 1).

Denote by p some prompt. Let $t_1, t_2, \ldots, t_n$ be new tokens generated autoregressively by some model M. We fix $n \geq 1$ and focus most of our analysis on the case where $n = 2$ as any finite n can be derived inductively. Also denote by $\mathbb{P}$ the probability distribution over $t_1, t_2, \ldots, t_n$ conditioned on some prompt p induced by the model M (Cotterell et al., 2023; Du et al., 2023). Let $\mathbb{P}^{\text{do}}$ be the model after intervention by one of the methods described previously.

Next-token patching. Let $t_1^\star, t_2^\star, \ldots, t_n^\star$ be the expected output for a given prompt p. For a fixed prompt, next-token patching defines the following patching metric

$$\text{NextToken}(p) := \text{KL}(\mathbb{P}[t_1|p] \;||\; \mathbb{P}^{\text{do}}[t_1|p]) + \quad (7)$$

$$\sum_{i=2}^{n} \text{KL}\left(\mathbb{P}[t_i|p, t^\star_{[:i]}] \;||\; \mathbb{P}^{\text{do}}[t_i|p, t^\star_{[:i]}]\right). \quad (8)$$

$\text{KL}(\mathbb{P} \;||\; \mathbb{P}^{\text{do}})$ denotes the KL-divergence between $\mathbb{P}$ and $\mathbb{P}^{\text{do}}$. Essentially, Equation 8 measures the effect of patching any given component on model M's next word prediction ability. Specifically, this metric captures a "local property" of M since in each summand, we assume that M has previously generated the correct tokens. To compute

Feature	Next-token Patching	Joint-token Patching
Granularity	Token	Sequence
Focus	Localized, stepwise effects	Global multi-token coherence
Computational Cost	Can compute exactly	High; requires approximations
Insight	Fine-grained, task-specific behavior	Broader token dependencies

Table 1: Features of next-token patching versus joint-token patching. The former can be seen as grokking the local token-level behavior of the model compared to the latter which can be viewed as uncovering global token dependencies.

the patching metric across the entire task distribution, we simply take the expectation over p: $\mathbb{E}_p[\text{NextToken}(p)]$.

Joint-token patching. As discussed previously, a language model can be thought of as a measure over the space of all sentences: $\mathbb{P}$. Joint-token patching directly measures the difference between the measure $\mathbb{P}$ (induced by M) and $\mathbb{P}^{\text{do}}$ (induced by an intervention on M) over the joint distribution of all n-tokens. Concretely, for a fixed prompt p,

$$\text{JointToken}(p) := \text{KL}(\mathbb{P}[t_1 \dots t_n|p] \parallel \mathbb{P}^{\text{do}}[t_1 \dots t_n|p])$$

By expanding the definition of $\text{KL}(\cdot \parallel \cdot)$, it is easy to see that for $n = 2$:

$$\text{JointToken}(p) = \text{KL}(\mathbb{P}[t_1|p] \parallel \mathbb{P}^{\text{do}}[t_1|p]) + \mathbb{E}_{t_1 \sim \mathbb{P}[t_1|p]}\left[\text{KL}(\mathbb{P}[t_2|p, t_1] \parallel \mathbb{P}^{\text{do}}[t_2|p, t_1])\right].$$

Note the expectation in the second summand. This is with respect to the token t_1 generated by the model without any intervention. The computation of this expectation is hard especially if n is large.

In both next-token and joint-token patching KL-divergence is used. This follows from the recommendation and positive results of Conmy et al. (2023), but one can technically use any other preferred metric. The key idea being that the metric should capture the performance decrease of the model after ablating an individual edge.

B.2 Noisy-to-Clean Patching

In Section 4 and 5, we perform noisy-to-clean patching. That is, for a model M, we run the model on a given example x and cache all of the activations across all of the edges. Then, we take another example x' which is associated with a different ground truth example. While the model runs on x', we patch in activations from x and check to see how much performance decreases with respect to the label associated with x'. In the case of attribution patching, we apply the same Taylor-expansion as above. Noisy-to-clean activation gives us a picture of the necessary components of the model.

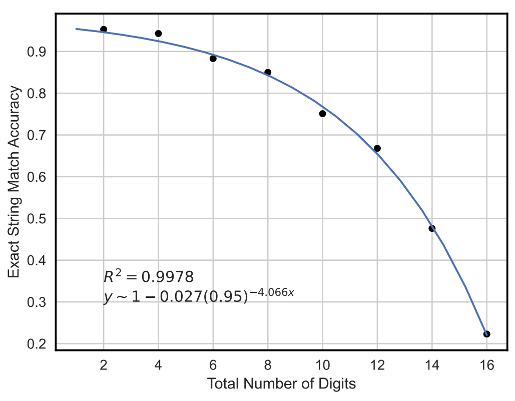

Figure 9: Exponential regression on exact string accuracy for equal-digit subtasks. That is, $o_1 = o_2$. The x-axis of the regression is the sum of the number of digits in both operands.

B.3 Clean-to-Clean Patching

In cases where there is not necessarily a ground-truth label, noisy-to-clean patching is not well-defined. Herein, we propose a new type of patching which we call clean-to-clean patching. This is applied in Section 6 and generally useful for circuit discovery where the output is some open-form generation for autoregressive language models. Let x be the input and $x^\star$ be the model's sampled response of x. We first run the model on x with padding tokens such that x and $x^\star$ have the same length. Similar to the procedure above, we cache all of the intermediate activations. Then, we run the model on $x^\star$ and patch in the appropriate activations from this "blanked" out x. The idea here is that we are finding the necessary components that exactly recover the model's response on this input, where the baseline is if the model had not generated anything at all. In the case of attribution patching, we apply the same Taylor-expansion as above.

C Within-Task Regressions

In Section 4, we argued that within an α-equivalent circuit family, performance decreases predictability.

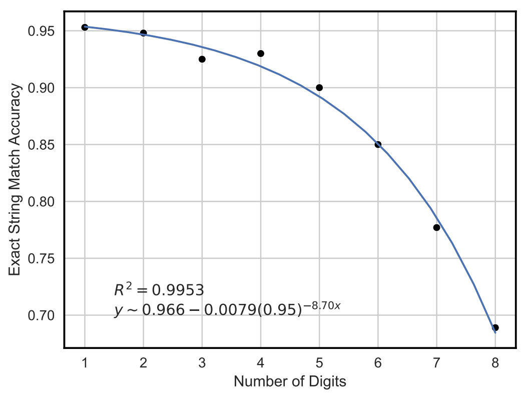

Figure 10: Exponential regression on exact string accuracy for one-digit subtasks. That is $o_2 = 1$. The x-axis of the regression is simply the number of digits in o_1 as o_2 is constant.

Herein, we perform a regression analysis to analyze this hypothesis. Specifically, we examine whether the subtask performances shown in Figure 4 can be predicted through an exponential regression using the performance of the $(1, 1)$ subtask. We perform an exponential regression since we expect error to be compounded. That is, for a $(2, 2)$ task, at least 4 $(1, 1)$ subtasks need to be computed. Thus, the error should be compounding on the order of 0.95^4. Likewise, a subtask that requires n, $(1, 1)$ subtask decompositions should require incur error on the order of $O(0.95^n)$. Therefore, consider a regression of the form $y = b - c(0.95)^{ax}$, where a, b, c are learnable parameters and x is the total number of digits across both operands, and ax is the total number of subtask decompositions. These regression results are shown in Figure 9 and Figure 10. We find that a strong R^2 is observed > 0.99.

D Reproducibility

Throughout the paper, we do not perform any finetuning or training. Rather, we directly evaluate the pretrained models. All of our experiments were conducted on two NVIDIA A100 80GB GPUs. Our codebase including the implementations of the proposed algorithms and figures can be found at `https://github.com/alansun17904/circuit-stability`. The models and their weights used as case study throughout the paper are loaded directly from the `transformer_lens` package, its documentation can be found at `https://transformerlensorg.github.io/TransformerLens/index.html`. We use all of the default hyperparameters and settings of the package.