Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR meets Machine Learning and Explanation

Do Transformers Learn What Theory Predicts? Knowledge Representation-Guided Mechanistic Verification via Causal Abstraction

Chang Lu[1] , **Renate A. Schmidt**[2] , **Yizheng Zhao**[3,*]
[1]Computational Biology and Biomedical Informatics Program, Yale University, New Haven, CT, USA
[2]Department of Computer Science, The University of Manchester, Manchester, UK
[3]State Key Laboratory of Novel Software Technology & School of Artificial Intelligence, Nanjing University, Nanjing, China
chang.lu@yale.edu, renate.schmidt@manchester.ac.uk, zhaoyz@nju.edu.cn

Abstract

Knowledge Representation (KR) formalisms provide precise specifications of algorithmic structure, yet it remains unclear whether gradient-trained neural networks implement these specifications even when they are theoretically expressible. Recent formal language theory tells us what Transformers *can* express—masked hard-attention Transformers recognize the star-free regular languages, equivalent to first-order logic with linear order ($\mathrm{FO}[<]$) and linear temporal logic (LTL)—but not what gradient-trained Transformers *will* learn.

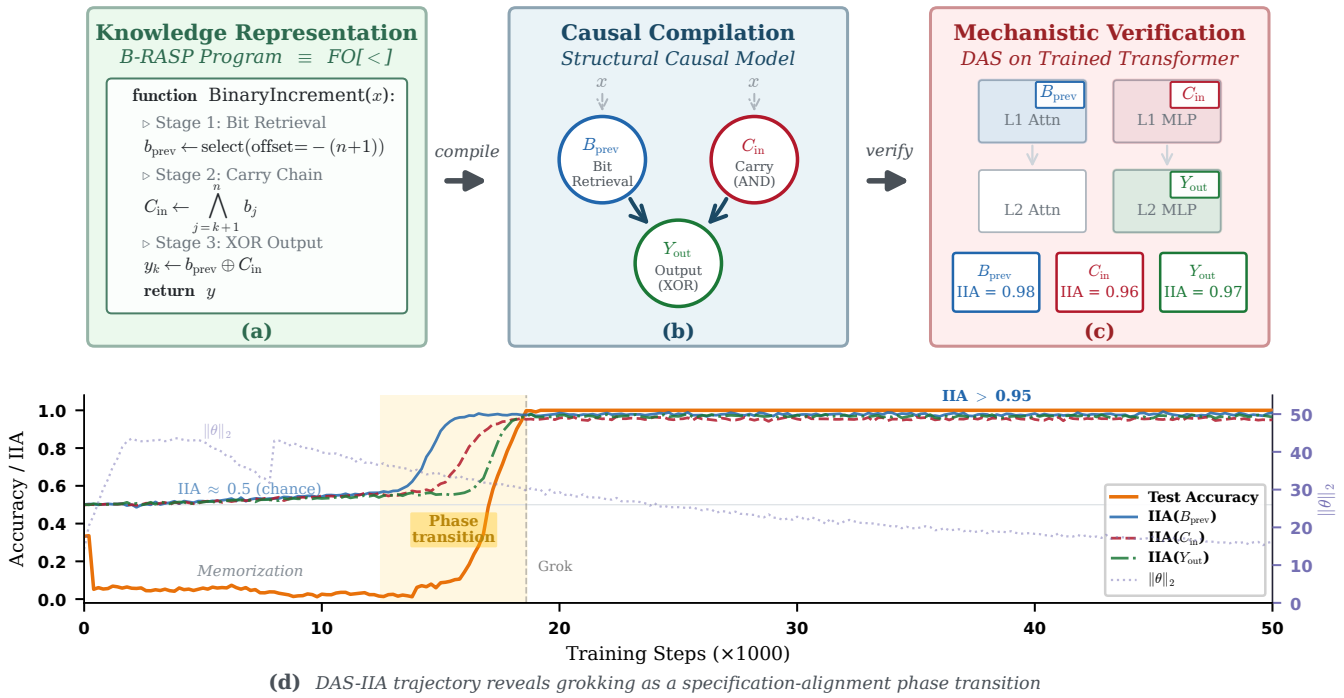
We ask whether trained networks actually implement the logical circuits that theory predicts, and propose Knowledge Representation-Guided Mechanistic Verification to find out. The idea is to compile a B-RASP specification into a Structural Causal Model whose variables correspond to prescribed logical operations, then use Distributed Alignment Search to obtain quantitative, falsifiable causal evidence for or against each operation's presence in the trained network. We instantiate this framework on a task of binary incrementfor which B-RASP prescribes an explicit three-stage circuit implementable by a two-layer Transformer with $O(\mathrm{poly}(n))$ parameters—exponentially fewer states than any fixed-precision recurrent or state space baselines require. The answer is affirmative: all three prescribed operations are faithfully encoded in dedicated neural subspaces, with wrong-specification controls at chance; Sparse Autoencoder decomposition independently recovers the same logical structure without supervision. Moreover, this structure is not acquired gradually: it emerges as a sharp phase transition during grokking, providing a specification-aligned progress measure that reveals *what* changes during the generalization transition, not merely *that* something changes. These results demonstrate that KR formalisms can serve not only as prescriptive specifications of what networks should compute, but also as falsifiable causal hypotheses that mechanistic interpretability tools can rigorously test, thereby bridging the gap between symbolic KR and neural computation.

1 Introduction

Formal language theory has recently established a striking connection between KR formalisms and neural network architectures (Baral and Giacomo 2015; Lieto 2021). Yang et al. (2024), building on the Restricted Access Sequence Processing Language (RASP) (Weiss, Goldberg, and Yahav 2021), introduced a Boolean restriction called B-RASP and proved that masked hard-attention Transformers (Vaswani et al. 2017) recognize the star-free regular languages, a class equivalent to first-order logic with linear order, $\mathrm{FO}[<]$ (McNaughton and Papert 1971), and LTL (Pnueli 1977). This result extends a growing body of work on the formal characterization of *inter alia* Transformer computation (Weiss, Goldberg, and Yahav 2021; Merrill and Sabharwal 2024; Strobl et al. 2024; Strobl et al. 2025). Though these equivalences tell us what Transformers *can* express, not what gradient-trained Transformers *will* learn: simplicity bias (Shah et al. 2020; Geirhos et al. 2020) and shortcut learning (Bhattamishra et al. 2023) may steer optimization toward heuristic solutions that reproduce correct outputs without implementing the principled algorithm that the theory predicts.

The gap becomes especially stark in the context of *succinctness* results, which have established a family of representational separations between Transformers and recurrent architectures (RNNs) (Bhattamishra et al. 2024; Sanford, Hsu, and Telgarsky 2023). Bergstraßer et al. (2026) recently proved that a fixed-precision Transformer can recognize the n-bit binary increment language with only polynomially many parameters, whereas any deterministic finite automaton (DFA)—and hence any fixed-precision RNNs (Elman 1990; Merrill et al. 2020) such as LSTM (Hochreiter and Schmidhuber 1997) and GRU (Cho et al. 2014) and any State Space Models (SSMs) (Gu, Goel, and Ré 2022) such as Mamba (Gu and Dao 2023)—requires exponentially many states (Nerode 1958). What makes this separation useful for empirical study is that the proof is *constructive*: the B-RASP formalism (Yang, Chiang, and Angluin 2024; Bergsträßer, Cotterell, and Lin 2026) provides an explicit program specifying a three-stage circuit—attention-based bit retrieval, carry-chain computation via iterated conjunction, and output via exclusive disjunction—that a two-layer Transformer can implement. This explicit specification opens the door to *mechanistic verification*: rather than merely showing that a Transformer can solve the task, one can test whether a trained Transformer internally implements the prescribed algorithm. Indeed, the Tracr compiler (Lindner et al. 2023) has shown that RASP programs (of which B-RASP is a subset) can be compiled into concrete Transformer weights, confirming the architectural feasibility of such implementations.

*Corresponding author



(d) *DAS-IIA trajectory reveals grokking as a specification-alignment phase transition*

Figure 1: Overview of Knowledge Representation-Guided Mechanistic Verification. **Panel (a):** a B-RASP program is compiled into a three-variable SCM. **(b):** DAS verifies each variable against the trained Transformer, producing per-component IIA scores. **(c):** tracking IIA across training checkpoints reveals a phase transition coinciding with grokking.

How can we verify whether a trained model actually implements this specification? This is fundamentally a question of mechanistic interpretability (Olah et al. 2020; Elhage et al. 2021), yet the standard tools face well-documented challenges. Attention weight visualization does not reliably indicate information flow or causal function (Jain and Wallace 2019; Wiegreffe and Pinter 2019). Activation patching (Meng et al. 2022; Wang et al. 2023) localizes important components by ablation, but can produce interpretability illusions at the subspace level (Makelov et al. 2024) and conflates sufficiency with necessity (Heimersheim and Nanda 2024). At the neuron level, polysemanticity (Elhage et al. 2022) makes single-neuron analysis unreliable (Sharkey et al. 2025): because neural networks pack more features into activation space than there are neuron dimensions (superposition), individual neurons typically encode mixtures of unrelated concepts rather than cleanly isolable variables. Verifying a formal prediction against a trained network therefore requires *quantitative, falsifiable causal evidence*.

We propose **KR-Guided Mechanistic Verification**, a framework that converts the B-RASP specification into a *testable causal claim*. The core idea is to compile a B-RASP program into a Structural Causal Model (SCM) (Pearl 2009): each B-RASP operation becomes a causal variable and the program's dataflow graph defines the causal arrows, yielding a three-variable SCM for n-bit binary increment whose variables correspond to bit retrieval, carry conjunction, and output exclusive disjunction. In this way, the compilation step transforms the question "*does the Transformer implement the B-RASP specification?*" into a precise causal abstraction problem (Geiger et al. 2021). We employ Distributed Alignment Search (DAS) (Geiger et al. 2024), an established causal verification method that uses gradient descent to find an orthogonal rotation of the neural activation space such that a low-dimensional subspace aligns with a given causal variable (Wu et al. 2023; Mueller et al. 2025; Wu, Geiger, and Millière 2025; Sun et al. 2025). Alignment quality is measured by Interchange Intervention Accuracy (IIA): when the aligned subspace is swapped between two inputs, IIA is the fraction of interventions under which the network's output agrees with the causal model's counterfactual prediction. As complementary evidence, we train Sparse Autoencoders (SAEs) (Templeton et al. 2024; Huben et al. 2024), following Marks et al. (2025) who integrated SAEs with causal circuit discovery, to examine whether the learned features correspond to the predicted logical operations. Figure 1 illustrates the complete pipeline from KR specification to mechanistic verification.

Beyond static verification of fully converged models, our framework also addresses a dynamic question: *when* during learning does a neural network transition from memorizing training examples to implementing a genuine logical circuit? We study this through the lens of grokking (Power et al. 2022), a phenomenon in which generalization appears suddenly after prolonged overfitting, and which has attracted substantial mechanistic and theoretical analysis (Nanda et al. 2023; Kumar et al. 2024; Lyu et al. 2024). Prior work tracks grokking with task-agnostic statistics such as weight

norm and effective rank, but these measures cannot detect whether the model has acquired the specific logical structure predicted by the KR specification—they signal that something is changing, not what is changing. We introduce the DAS-IIA trajectory as a *specification-aligned progress measure* that fills this gap: by evaluating causal alignment at every training checkpoint, it reveals an abrupt transition from chance level (IIA ≈ 0.5) to near-perfect specification alignment (IIA > 0.95) that coincides precisely with the onset of generalization. We make the following contributions:

1. We propose KR-guided mechanistic verification, a novel framework that systematically compiles B-RASP specifications into SCMs and verifies their neural implementation via DAS. This establishes a principled bridge from symbolic KR to neural mechanistic interpretability.
2. Under strict parameter matching, we demonstrate that Transformers achieve perfect generalization on n-bit binary increment via grokking, while all RNN and SSM baselines fail, providing controlled empirical confirmation of the theoretical succinctness separation.
3. DAS yields IIA ≥ 0.95 for all three B-RASP causal variables, and SAE decomposition independently recovers monosemantic features corresponding to the predicted logical operations, providing convergent evidence from two methodologically independent analyses.
4. DAS-IIA trajectory reveals that specification alignment emerges as a sharp phase transition during grokking, offering a new, specification-aligned progress measure that is more informative than weight norm or effective rank.

These results establish KR formalisms as operationally falsifiable specifications that guide and evaluate mechanistic interpretability of neural computation.

Guide for the KR readers who might be less familiar with the Machine Learning (ML) side-methods. This paper's contributions map onto classical KR concepts as follows: B-RASP serves as a *specification* language (equivalent to FO[<]/LTL); the B-RASP-to-SCM step is *knowledge compilation*, transforming one formalism into another to enable causal intervention; DAS performs *verification*, testing whether a system satisfies the specification; and the succinctness separation (Theorem 1) is a classical *representational trade-off* result in the tradition of descriptive complexity. Expanded background is provided in Section 2.

2 Preliminaries

2.1 Transformers and B-RASP

We briefly define the Transformer variant relevant to the formal characterization; readers familiar with masked Transformers may proceed to Theorem 1.

Let Σ be a finite alphabet and Σ^* the set of all finite strings over Σ. A *language* is a set $L \subseteq \Sigma^*$. A *Masked Hard-Attention Transformer* (MHAT) with ℓ layers, H heads per layer, and hidden dimension d processes an input $x = x_1 \cdots x_T \in \Sigma^*$ of length T as follows. Each token x_t ($t \in \{1, \ldots, T\}$) is mapped to an initial representation $h_t^{(0)} \in \mathbb{R}^d$ via a learned embedding (a lookup table that maps each symbol to a d-dimensional real vector) combined with positional information. At layer $l \in \{1, \ldots, \ell\}$, each head $a \in \{1, \ldots, H\}$ has learned projection matrices $W_Q^{(l,a)}$, $W_K^{(l,a)}$, $W_V^{(l,a)} \in \mathbb{R}^{d \times d}$.[1] The head computes a query $q = W_Q^{(l,a)} h_t^{(l-1)}$ at position t and, for each candidate source $s \in \{1, \ldots, t\}$, a key $\kappa_s = W_K^{(l,a)} h_s^{(l-1)}$. It selects $s^* = \arg\max_{s \leq t} q^\top \kappa_s$, where $s \leq t$ enforces causal masking and a fixed rule breaks ties. It retrieves $W_V^{(l,a)} h_{s^*}^{(l-1)}$; the outputs of all H heads are concatenated and projected by a learned matrix $W_O^{(l)} \in \mathbb{R}^{d \times d}$, then added to the current representation $h_t^{(l-1)}$ via a residual connection. A position-wise multi-layer perceptron (MLP), consisting of two affine transformations separated by a nonlinear activation, is then applied and added via a second residual connection, yielding the updated representation $h_t^{(l)}$. Residual connections ensure that information computed at earlier layers remains directly accessible at later layers. We write $\mathrm{MHAT}(\ell, H, d)$ for this model class.

RASP abstracts these components into three primitives: *select*, which produces a Boolean attention pattern specifying which source positions a given target position attends to; *aggregate*, which retrieves and combines values from the selected positions; and *pointwise* operations computed by the MLP. B-RASP restricts all intermediate values to Booleans (i.e., every select, aggregate, and pointwise operation maps to $\{0, 1\}$), yielding a tight expressiveness characterization.

Theorem 1 (Yang et al. 2024)**.** *L is recognized by some $\mathrm{MHAT}(\ell, H, d)$ iff L is star-free ($\mathrm{FO}[<]$-definable).*

Theorem 1 applies to hard attention; *softmax* attention replaces the argmax with a normalized exponential weighting, $\mathrm{softmax}(q^\top \kappa_s) = \exp(q^\top \kappa_s) / \sum_{s' \leq t} \exp(q^\top \kappa_{s'})$, producing a weighted average over all source positions rather than selecting a single one. However, when attention logits are large, softmax concentrates on the argmax and approximates hard attention; weight decay encourages low-rank weight matrices that sharpen attention patterns (Lyu et al. 2024). No formal guarantee extends the theorem to softmax models; one of the goals of our empirical investigation (Section 5) is precisely to test whether softmax Transformers, despite this theoretical gap, nonetheless learn circuits that are functionally equivalent to the B-RASP specification.

2.2 The n-bit Binary Increment Task

Fix an integer $n \geq 1$. The language L_n of n-bit binary increment is defined over $\Sigma = \{0, 1, \#\}$. An input has the form $b_1 \cdots b_n \# y_1 \cdots y_n$, where $b_1 \cdots b_n$ is an n-bit binary number (most-significant bit (MSB) first) and $y_1 \cdots y_n$ is the successor of $b_1 \cdots b_n$ modulo 2^n. The task is next-token prediction: at each output position $k \in \{1, \ldots, n\}$, the model predicts y_k given the prefix $b_1 \cdots b_n \# y_1 \cdots y_{k-1}$.

[1] Following the MHAT formalization of Yang et al. (2024), each head uses full $d \times d$ matrices to simplify the expressiveness proof. Our model (Section 4.2) uses the standard multi-head convention with per-head dimension $d_k = d/H$. Since the per-head parameterization is a strictly less expressive special case of the full-rank formulation, the expressiveness guarantee of Theorem 1 carries over.

Succinctness. The exponential separation previewed in Section 1 rests on the Myhill–Nerode theorem (Nerode 1958): any DFA for L_n needs $\Omega(2^n)$ states, whereas a Unique Hard Attention Transformer (UHAT) (Hahn 2020)—an MHAT variant with deterministic tie-breaking—recognizes L_n with only $O(\mathrm{poly}(n))$ parameters (Bergsträßer, Cotterell, and Lin 2026). Since fixed-precision (finite bit) RNNs and SSMs are bounded by finite-state computation, this yields an exponential succinctness separation.

B-RASP specification. The B-RASP program for L_n prescribes three operations per output position k (Yang, Chiang, and Angluin 2024; Bergsträßer, Cotterell, and Lin 2026):

$$B_{\mathrm{prev}}(k) := b_k \quad \text{(retrieve } k\text{-th input bit)}, \tag{1}$$

$$C_{\mathrm{in}}(k) := \textstyle\bigwedge_{j=k+1}^{n} b_j \quad \text{(carry: AND of lower bits)}, \tag{2}$$

$$Y_{\mathrm{out}}(k) := B_{\mathrm{prev}}(k) \oplus C_{\mathrm{in}}(k) \quad \text{(output: XOR)}, \tag{3}$$

where $\oplus$ denotes exclusive disjunction and $\bigwedge$ iterated conjunction. We adopt the standard convention that an empty conjunction evaluates to $\top$ (True); hence $C_{\mathrm{in}}(n) = 1$, so the least-significant bit always flips, as expected for $x+1$. In B-RASP terms, B_{prev} uses *select* to attend to position k of the input (at a fixed relative offset of $-(n+1)$), followed by *aggregate* to copy the bit value; this corresponds to a single attention head. C_{in} and Y_{out} are *pointwise* Boolean functions computed in the MLP. These three causal variables constitute the specification that our framework verifies.[2]

2.3 Causal Models and Causal Abstraction

A *Structural Causal Model* (SCM) (Pearl 2009) is a tuple $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathcal{F}, P(\mathbf{U}))$. Here $\mathbf{V} = \{V_1, \ldots, V_m\}$ is a set of m endogenous variables and $\mathbf{U}$ is a set of exogenous variables. $\mathcal{F} = \{f_1, \ldots, f_m\}$ is the set of structural equations $V_i := f_i(\mathrm{pa}(V_i), U_i)$ for each $i \in \{1, \ldots, m\}$, with $\mathrm{pa}(V_i)$ the parents of V_i and $U_i \in \mathbf{U}$ the corresponding exogenous input. $P(\mathbf{U})$ is a joint distribution over $\mathbf{U}$. An *intervention* $\mathrm{do}(V_i{=}v)$ replaces V_i's equation with the constant v, leaving all others unchanged.

Intuitively, to test whether a neural network $\mathcal{N}$ implements a high-level SCM $\mathcal{H}$, one checks that surgically swapping a specific part of the network's internal state between two inputs changes the output in exactly the way the causal model predicts. Formally (Geiger et al. 2021), one defines an *alignment* $\tau\colon \mathbf{V}_{\mathcal{H}} \to 2^{\mathbf{V}_{\mathcal{N}}}$ that maps each high-level variable to a set of neural representations (because a concept may persist across layers via residual connections), where $\mathbf{V}_{\mathcal{H}}$ and $\mathbf{V}_{\mathcal{N}}$ denote the endogenous variables of $\mathcal{H}$ and $\mathcal{N}$ respectively. An *interchange intervention* on $\mathcal{N}$ w.r.t. $V_i \in \mathbf{V}_{\mathcal{H}}$ runs $\mathcal{N}$ on a *base* input x_b but replaces the activations in $\tau(V_i)$ with those from a *source* x_s. *Interchange Intervention Accuracy* (IIA) measures the resulting agreement over a held-out evaluation dataset $\mathcal{D}$ of base–source pairs:

$$\mathrm{IIA}(V_i) = \frac{1}{|\mathcal{D}|} \sum_{(x_b, x_s) \in \mathcal{D}} \mathbf{1}\big[\mathcal{N}\big(\widetilde{h}(x_b, x_s; \tau(V_i))\big) = \mathcal{H}\big(\mathrm{do}(V_i{:=}V_i(x_s))\big)(x_b)\big], \tag{4}$$

[2] A symmetric analysis of n-bit binary decrement, whose borrow variable replaces the carry, produces consistent results.

where $\mathbf{1}[\cdot]$ denotes the indicator function, $\widetilde{h}$ the intervened activations, and $V_i(x_s)$ the value of V_i when $\mathcal{H}$ processes x_s. The left-hand side of the comparison, $\mathcal{N}(\widetilde{h})$, is the network's output when its internal activations are surgically replaced; the right-hand side, $\mathcal{H}(\mathrm{do}(V_i{:=}v))(x_b)$, is the output of the SCM $\mathcal{H}$ when evaluated on input x_b with the intervention $V_i{:=}v$. IIA $= 1$ indicates *exact* causal abstraction—the neural network perfectly implements the high-level SCM; for binary variables, chance is 0.5. In practice, IIA < 1 reflects *approximate* causal abstraction (Geiger et al. 2021), where the alignment holds for most but not all interventions. We write $\mathrm{IIA}(V_i)$ for the position-averaged IIA of variable V_i, $\mathrm{IIA}(V_i, k)$ when restricting to output position k, $\mathrm{IIA}(V_i, c)$ when specifying the Transformer component c, and $\mathrm{IIA}(V_i;\, \theta_p)$ for IIA evaluated at training checkpoint θ_p.

2.4 Distributed Alignment Search

The alignment τ is unknown a priori, since causal variables need not correspond to individual neurons (Section 1). *Distributed Alignment Search* (DAS) (Geiger et al. 2024) addresses this by learning an orthogonal matrix $R \in \mathrm{O}(d)$ that rotates the activation space at a chosen Transformer component. Given a target subspace dimension r, we write $R_{:,1:r} \in \mathbb{R}^{d \times r}$ for the submatrix formed by the first r columns of R; these columns span the alignment subspace for V_i. DAS implements the intervened activation $\widetilde{h}$ of Eq. 4 as

$$\widetilde{h} = h^{(b)} + R_{:,1:r}\big(R_{:,1:r}^{\top} h^{(s)} - R_{:,1:r}^{\top} h^{(b)}\big), \tag{5}$$

where $h^{(b)}$ and $h^{(s)}$ are the activations of the base and source inputs at the chosen component. This operation replaces the component of $h^{(b)}$ that lies within the r-dimensional alignment subspace with the corresponding component from $h^{(s)}$, leaving all other directions unchanged. R is optimized by gradient descent to maximize IIA (Eq. 4). A high IIA after convergence confirms that the causal variable is faithfully encoded in an r-dimensional subspace.

2.5 Sparse Autoencoders

A *Sparse Autoencoder* (SAE) (Templeton et al. 2024) provides a complementary, unsupervised decomposition of activations. Given $h \in \mathbb{R}^d$, an SAE with dictionary size D ($D \gg d$) computes a sparse code $z \in \mathbb{R}^D$ and a reconstruction $\hat{h} \in \mathbb{R}^d$. It uses a learned encoder $W_{\mathrm{enc}} \in \mathbb{R}^{D \times d}$, a decoder $W_{\mathrm{dec}} \in \mathbb{R}^{d \times D}$, and biases $\beta_{\mathrm{enc}} \in \mathbb{R}^D$, $\beta_{\mathrm{dec}} \in \mathbb{R}^d$:

$$z = \mathrm{TopK}\big(W_{\mathrm{enc}}(h - \beta_{\mathrm{dec}}) + \beta_{\mathrm{enc}}\big),\ \hat{h} = W_{\mathrm{dec}} z + \beta_{\mathrm{dec}}, \tag{6}$$

where TopK retains the K_{sae} largest activations for a fixed sparsity level $K_{\mathrm{sae}} \ll D$, zeroing the rest. The SAE minimizes $\|h - \hat{h}\|_2^2$. A feature z_j (the j-th entry of z) is called *monosemantic* if it activates selectively for a single interpretable concept (Elhage et al. 2022). Where DAS tests a *given* causal hypothesis, SAE features are discovered without supervision (Marks et al. 2025). Section 3.2 describes how we combine the two methods.

2.6 Grokking and Progress Measures

Grokking (Power et al. 2022; Carvalho et al. 2025) is delayed generalization long after training accuracy has saturated, often spanning thousands of gradient steps. Let θ denote the model parameters. Existing progress measures for this transition include the *excluded loss* (Nanda et al. 2023), the L_2 weight norm $\|\theta\|_2$, and the effective rank of weight matrices (Kumar et al. 2024; Lyu et al. 2024).[3] Section 3 introduces a specification-aligned alternative.

3 KR-Guided Mechanistic Verification

Our framework treats the B-RASP specification as a *falsifiable causal hypothesis* about a trained Transformer's internal computation. It proceeds in three steps: (1) compile the B-RASP program into an SCM whose variables and causal arrows mirror the program's dataflow (Section 3.1), (2) use DAS to verify whether each causal variable is faithfully encoded in Transformer's activation space (Section 3.2), and (3) track alignment across training checkpoints as a specification aligned progress measure for grokking (Section 3.3).

3.1 Compiling B-RASP into an SCM

Let $\mathcal{P}$ be a B-RASP program with m variables $V_1, \ldots, V_m$, each defined by a Boolean function of other program variables and/or the input tokens. We compile $\mathcal{P}$ into an SCM $\mathcal{H}_\mathcal{P} = (\mathbf{V}, \mathbf{U}, \mathcal{F}, P(\mathbf{U}))$ as follows.

Definition 1 (B-RASP-to-SCM compilation)**.** *Given a B-RASP program $\mathcal{P}$ with variables $V_1, \ldots, V_m$:*

1. ***Endogenous variables.*** *Set $\mathbf{V} = \{V_1, \ldots, V_m\}$, inheriting the Boolean domain $\{0, 1\}$ from B-RASP.*
2. ***Exogenous variables.*** *Let $\mathbf{U}$ be the set of input symbols consumed by $\mathcal{P}$, with $P(\mathbf{U})$ the data distribution.*
3. ***Structural equations.*** *For each V_i, let $\mathrm{pa}(V_i) \subseteq \mathbf{V}$ be the set of program variables in the definition of V_i, and let $\mathbf{U}_i \subseteq \mathbf{U}$ be the exogenous variables it reads. Set f_i to the Boolean function prescribed by $\mathcal{P}$, so that $V_i := f_i(\mathrm{pa}(V_i), \mathbf{U}_i)$. Collect $\mathcal{F} = \{f_1, \ldots, f_m\}$.*
4. ***Causal graph.*** *The directed acyclic graph (DAG) $G = (\mathbf{V} \cup \mathbf{U}, E)$ has an edge $V_j \to V_i$ for each $V_j \in \mathrm{pa}(V_i)$ and an edge $U_q \to V_i$ for each $U_q \in \mathbf{U}_i$.*

Proposition 1. *For any B-RASP program $\mathcal{P}$, the compilation in Definition 1 yields a valid SCM: the causal graph G is acyclic and each structural equation is deterministic.*

In the standard SCM formulation (Section 2.3), each structural equation reads a single exogenous variable U_i. Here, each equation may read multiple distinct exogenous inputs $\mathbf{U}_i \subseteq \mathbf{U}$ (e.g., C_{in} reads all n input bits), reflecting the B-RASP program's data dependencies. Because every endogenous variable is Boolean, the chance-level IIA is exactly 0.5. The compilation is well-defined for any B-RASP program: since B-RASP variables are defined by acyclic dataflow (Yang, Chiang, and Angluin 2024), the resulting causal graph G is guaranteed to be a DAG, and the structural equations are fully determined by the program text.

Instantiation for L_n. Applying Def. 1 to Eqs. 1–3 yields a three-variable SCM $\mathcal{H}_{L_n}$ with $\mathbf{V} = \{B_{\mathrm{prev}}, C_{\mathrm{in}}, Y_{\mathrm{out}}\}$ and $\mathbf{U} = \{b_1, \ldots, b_n\}$. The structural equations are Eqs. 1–3 themselves.

An interchange intervention on, say, B_{prev} replaces its activation in a base run with that from a source run, testing whether the Transformer derives Y_{out} from its B_{prev} representation via the XOR of Eq. 3, rather than through an alternative computational path.

3.2 Verification via DAS

Given a compiled SCM $\mathcal{H}_\mathcal{P}$ and a trained Transformer $\mathcal{N}$, we verify whether $\mathcal{N}$ faithfully implements each causal variable $V_i \in \mathbf{V}$.

Candidate components. A Transformer with ℓ layers offers a set of candidate components

$$\mathcal{C} = \big\{(l, \mathrm{type}) : l \in \{1, \ldots, \ell\},\ \mathrm{type} \in \{\texttt{attn}, \texttt{mlp}\}\big\}, \tag{7}$$

where `attn` and `mlp` denote the attention and feed-forward sub-layers of layer l, respectively. Concretely, let $a^{(l)} = h^{(l-1)} + \mathrm{Attn}^{(l)}(h^{(l-1)})$ denote the residual-stream vector after the attention sub-layer of layer l; the activation probed at component $(l, \texttt{attn})$ is $a^{(l)}$, and the activation at $(l, \texttt{mlp})$ is $a^{(l)} + \mathrm{MLP}^{(l)}(a^{(l)})$, i.e. the full layer output $h^{(l)}$.

Verification procedure. For each $V_i \in \mathbf{V}$ and each $c \in \mathcal{C}$, we apply DAS (Section 2.4) at c with target dimension r and compute $\mathrm{IIA}(V_i, c)$ via Eq. 4. We say V_i is *faithfully encoded* at c if $\mathrm{IIA}(V_i, c) > 0.95$, well above the chance level of 0.5 for binary variables and conservative enough to allow for statistical noise in finite evaluation sets. This threshold follows the convention in the DAS literature (Geiger et al. 2024; Wu et al. 2023); our conclusions are robust to the exact choice: at 0.90, every variable at every bit width passes, while at 0.99 only the easiest case (B_{prev} at n=8) passes. Formal one-sided t-tests with Bonferroni correction (30 tests, α_{corr}=0.0017) are reported in the supplementary material. The specification $\mathcal{H}_\mathcal{P}$ is *verified* when every V_i achieves $\mathrm{IIA}(V_i, c_i^*) > 0.95$. For each V_i, we record the best-scoring component $c_i^* = \arg\max_{c \in \mathcal{C}} \mathrm{IIA}(V_i, c)$.

Structural predictions. The B-RASP primitives underlying each variable yield testable predictions about *where* it is encoded. B_{prev} involves *select* and *aggregate* (attention primitives), predicting $c^*_{B_{\mathrm{prev}}} \in \{(l, \texttt{attn}) : l \in \{1, \ldots, \ell\}\}$. C_{in} and Y_{out} are pointwise Boolean operations, predicting $c^*_{C_{\mathrm{in}}}, c^*_{Y_{\mathrm{out}}} \in \{(l, \texttt{mlp}) : l \in \{1, \ldots, \ell\}\}$. Section 5 tests these predictions against trained models.

SAE cross-validation. At each component c_i^*, we train an SAE (Section 2.5) and examine whether a monosemantic feature z_j aligns with V_i. Agreement between DAS (supervised, given the causal hypothesis) and an SAE feature (unsupervised) constitutes convergent evidence. The verification procedure above applies to a single converged model; we next extend it to the training trajectory.

[3] Effective rank measures the entropy of the normalized singular value distribution of a weight matrix; it decreases as the matrix becomes more low-rank (Kumar et al. 2024).

3.3 Specification-Aligned Progress Measure

The progress measures (Sec. 2.6) capture statistical properties of the parameters but do not indicate whether the model has acquired any particular algorithmic structure. We define a measure that directly tracks how faithfully the model implements the B-RASP causal model throughout training.

Definition 2 (DAS-IIA trajectory)**.** *Let $\theta_1, \theta_2, \ldots, \theta_C$ be a sequence of C training checkpoints. For each checkpoint θ_p ($p \in \{1, \ldots, C\}$) and each causal variable $V_i \in \mathbf{V}$, let* $\mathrm{IIA}(V_i; \theta_p)$ *denote the IIA obtained by running DAS at the best component c_i^* (Section 3.2) on the model with parameters θ_p. The* DAS-IIA trajectory *is the family of curves $\{p \mapsto \mathrm{IIA}(V_i; \theta_p)\}_{V_i \in \mathbf{V}}$. We define the* minimum-variable IIA *as*

$$\mathrm{IIA}_{\min}(\theta_p) \;=\; \min_{V_i \in \mathbf{V}} \mathrm{IIA}(V_i; \theta_p). \tag{8}$$

In practice, c_i^* is determined from the converged model and held fixed across all checkpoints; we verify this choice by confirming that the same component achieves maximal IIA at every post-grokking checkpoint (confirmed across all 5 seeds and all 500 post-grokking checkpoints). The trajectory has a direct mechanistic reading. IIA ≈ 0.5 at a given checkpoint means the model does not yet encode V_i in a linearly accessible subspace (the chance level for binary variables), while IIA ≈ 1.0 means it faithfully implements V_i. A sharp rise from 0.5 to 1.0 signals a *phase transition* from memorization to specification-aligned computation (we use this term in a phenomenological sense, referring to a rapid, threshold-like change in the order parameter $\mathrm{IIA}_{\min}$, following its established usage in the grokking literature (Power et al. 2022)).

Unlike $\|\theta\|_2$ or effective rank, which summarize global parameter statistics, $\mathrm{IIA}_{\min}$ directly tracks *whether the model has acquired the logical structure predicted by the KR specification*, thereby pinpointing when the transition from memorization to specification-aligned computation occurs.

4 Experimental Setup

This section describes the data, models, training procedure, and verification configurations used to instantiate the framework on the n-bit binary increment language L_n.

4.1 Task and Data

We evaluate on L_n for bit widths $n \in \{8, 12, 16, 20, 24\}$. Each instance is a string $b_1 \cdots b_n \# y_1 \cdots y_n$ of total length $T = 2n + 1$, where $b_1 \cdots b_n$ encodes an n-bit integer in most-significant-bit-first order and $y_1 \cdots y_n$ is its successor modulo 2^n. The task is autoregressive next-token prediction: at each output position $k \in \{1, \ldots, n\}$, the model predicts y_k given the prefix $b_1 \cdots b_n \# y_1 \cdots y_{k-1}$.

Dataset construction. For a given n, the complete dataset consists of all 2^n input–output pairs. We randomly partition this set into a training set of size $\lfloor \alpha \cdot 2^n \rfloor$ and a test set containing the remainder, using a train fraction $\alpha = 0.5$ for the main experiments. For $n \leq 16$, the full 2^n pairs are enumerable ($2^{16} = 65{,}536$); for $n \in \{20, 24\}$, we uniformly sample 50,000 pairs as the evaluation pool and apply the same $\alpha = 0.5$ split. To assess sensitivity to data fraction, we additionally report results for $\alpha \in \{0.3, 0.7\}$ at $n = 16$.

Metrics. We report two metrics: *per-position accuracy*, the fraction of correctly predicted output bits across all positions and test instances, and *per-sequence accuracy*, the fraction of test instances for which all n output bits are correct. Perfect generalization requires per-sequence accuracy of 1.0.

4.2 Model Architectures

Transformer. Our primary model is a decoder-only Transformer with $\ell = 2$ layers, $H = 2$ attention heads per layer, and hidden dimension $d = 128$. The architecture uses learned token embeddings and learned positional embeddings with maximum sequence length $T_{\max} = 2 \cdot 24 + 1 = 49$ (shared across all n). Attention is standard softmax with causal (left-to-right) masking, consistent with the $s \leq t$ constraint of the MHAT class (Section 2.1). Each layer consists of multi-head self-attention followed by a two-layer MLP with Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel 2016) and hidden dimension $d_{\mathrm{mlp}} = 4d = 512$, i.e. $\mathrm{MLP}(x) = W_2\,\mathrm{GELU}(W_1 x + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$ with bias terms in both layers. Residual connections and layer normalization ($\epsilon = 10^{-5}$) are applied in the pre-norm configuration. Attention logits are scaled by $1/\sqrt{d/H}$. The output head is a linear projection from $\mathbb{R}^d$ to the vocabulary logits ($|\Sigma| = 3$), without bias and without weight tying to the input embedding. No dropout is applied; regularization comes exclusively from weight decay (Section 4.3). All weights are initialized with PyTorch defaults (Kaiming uniform for linear layers (He et al. 2015)). This architecture is intentionally compact: two layers suffice to realize the three B-RASP stages (B_{prev} via attention, C_{in} and Y_{out} via MLPs), and $d = 128$ aligns with the standard configuration in the grokking literature (Power et al. 2022). Two layers is the minimum depth because Y_{out} depends on C_{in}, requiring separate MLP stages; a depth ablation confirms that ℓ=1 fails to realize the specification ($\mathrm{IIA}_{\min} = 0.61$), while ℓ=2 achieves full verification ($\mathrm{IIA}_{\min} = 0.96$).

Comparison with RNN and SSM Baselines. We train an LSTM (Hochreiter and Schmidhuber 1997) and a GRU (Cho et al. 2014) under strict parameter matching: hidden dimension of each RNN is chosen so that its total parameter count equals that of the Transformer to within 1%. Concretely, let P_T denote the Transformer's total parameter count. For each RNN, we choose the largest hidden dimension d_h whose total parameter count P_{rec} satisfies $P_{\mathrm{rec}} \leq P_T$.

To cover the recent class of SSMs, we train a Mamba (Gu and Dao 2023) baseline under the same parameter-matching protocol. Although Mamba's selective scan mechanism introduces input-dependent gating, its state dimension is fixed at initialization and does not grow with the input length; under our parameter-matching protocol, this yields a finite-state system under fixed-precision arithmetic whose total number of reachable configurations is bounded; $\Omega(2^n)$ state-complexity lower bound applies.

The Transformer has $P_T = 403{,}840$ parameters; matched hidden dimensions are $d_h = 223$ (LSTM, 400,957 params), $d_h = 258$ (GRU, 402,483 params), and $d_h = 239$ (Mamba,

403,674 params). Despite parameter matching, the two families face fundamentally different complexity regimes: the Transformer's $O(\mathrm{poly}(n))$-parameter B-RASP solution (Theorem 1) suffices for all n, whereas any fixed-precision recurrent network or SSM requires $\Omega(2^n)$ states for L_n.

4.3 Training Procedure

All models minimize next-token cross-entropy on the output positions $y_1, \ldots, y_n$ using AdamW (Loshchilov and Hutter 2019) (lr 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.98)$, weight decay $\lambda = 1.0$), following the grokking recipe of Power et al. (2022) and Nanda et al. (2023). We use batch size 512, linear warmup over 500 steps, and cosine decay to 0; no gradient clipping is applied. Training runs for 100,000 gradient steps, a budget sufficient to observe grokking across all tested n. Each configuration (model type $\times$ bit width $\times$ train fraction) is run with 5 independent random seeds; we report mean $\pm$ standard deviation.

Checkpoints and grokking protocol. We save checkpoints every 200 steps ($C = 500$ per run), with finer resolution (50 steps) near the grokking transition. The combination of small training set ($\alpha = 0.5$), strong weight decay ($\lambda = 1.0$), and extended training (100,000 steps) reliably induces grokking. The *memorization step* is the first step at which train per-sequence accuracy reaches 1.0; the *grokking step* is the first step at which test per-sequence accuracy exceeds 0.99; the delay is the *grokking gap*.

4.4 DAS Configuration

We apply DAS (Section 2.4) to verify whether the trained Transformer $\mathcal{N}$ faithfully encodes each causal variable $V_i \in \{B_{\mathrm{prev}}, C_{\mathrm{in}}, Y_{\mathrm{out}}\}$ of the compiled SCM $\mathcal{H}_{L_n}$ (Section 3.1).

Candidate components. Following the search procedure of Section 3.2, DAS is applied at every component in $\mathcal{C} = \{(l, \mathrm{type}) : l \in \{1, 2\}, \mathrm{type} \in \{\texttt{attn}, \texttt{mlp}\}\}$, giving four candidate components (L1-attn, L1-MLP, L2-attn, L2-MLP). For each variable V_i and each component $c \in \mathcal{C}$, we train a separate rotation R_c and report $\mathrm{IIA}(V_i, c)$. The best component $c_i^* = \arg\max_{c \in \mathcal{C}} \mathrm{IIA}(V_i, c)$ identifies where V_i is most faithfully encoded.

Subspace dimension. We use a target subspace dimension $r = 1$ for all causal variables, reflecting the fact that B_{prev}, C_{in}, and Y_{out} are binary-valued.

Position-level intervention. Because the B-RASP variables are defined per output position (Eqs. 1–3), we perform DAS interventions at each position $k \in \{1, \ldots, n\}$ independently, sampling base–source pairs uniformly from inputs where $V_i(k)$ differs. Non-target variables are not constrained; uniform sampling ensures that IIA reflects the causal role of V_i. The reported IIA is averaged over positions: $\mathrm{IIA}(V_i) = \frac{1}{n} \sum_{k=1}^{n} \mathrm{IIA}(V_i, k)$.

DAS training and trajectory analysis. For each (variable, component, position) triple, we train $R \in \mathrm{O}(d)$ using Adam (β_1=0.9, β_2=0.999, lr 10^{-3}) for 10 epochs on 2,000 base–source pairs (frozen Transformer weights), and evaluate on 1,000 held-out pairs; IIA saturates within the first 5 epochs in all configurations, and extending training to 20 epochs yields no further improvement. Although each position learns an independent rotation, the resulting directions $R_{:,1}$ are highly consistent across positions (mean pairwise cosine > 0.93 at n=16), confirming that the model uses a shared encoding direction for each causal variable. For the DAS-IIA trajectory (Def. 2), we re-train R at each checkpoint θ_p ($p \in \{1, \ldots, C\}$) at the best component c_i^*, yielding $C \times |\mathbf{V}| = 1{,}500$ runs per seed, each completing in seconds.

Negative controls. We construct two wrong-SCM variants by replacing the XOR in Y_{out} (Eq. 3) with AND ($\mathcal{H}_{\mathrm{AND}}$) or OR ($\mathcal{H}_{\mathrm{OR}}$), keeping others unchanged. We also run DAS with the correct SCM on a randomly initialized Transformer.

4.5 SAE Configuration

At each best component c_i^* identified by DAS, we train an SAE (Section 2.5) to obtain an unsupervised decomposition of the activations, following the integration of SAEs with causal circuit discovery by Marks et al. (2025).

Architecture and training. We use the TopK SAE variant (Eq. 6) with dictionary size $D = 8d = 1{,}024$ (8$\times$ expansion) and sparsity level $K_{\mathrm{sae}} = 8$; TopK provides exact sparsity control and avoids tuning an L_1 penalty coefficient. The SAE is trained on activations collected from 20,000 forward passes through the converged Transformer, using Adam with learning rate 3×10^{-4}, batch size 256, for 50 epochs, minimizing $\|h - \hat{h}\|_2^2$. Decoder columns are renormalized to unit norm after each gradient step.

Reconstruction quality. We report coefficient of determination $R^2 = 1 - \|h - \hat{h}\|_2^2 / \|h - \bar{h}\|_2^2$ on a held-out set, where $\bar{h}$ is the sample mean of the activations, to verify that SAE captures sufficient variance for reliable feature analysis.

Feature–variable alignment. For each feature z_j and variable V_i, we compute the Pearson correlation $\rho(z_j, V_i)$ and the classification accuracy of $z_j > 0$ as a predictor for $V_i = 1$. A feature is *monosemantic* w.r.t. V_i if $|\rho| > 0.9$ and classification accuracy ≥ 0.95; these conservative thresholds ensure that only features with strong univariate correspondence to a single causal variable are reported.

DAS–SAE convergence. We measure the cosine similarity between the DAS direction $R_{:,1}$ and the decoder column $\mathbf{w}_j \in \mathbb{R}^d$ of the best-matching SAE feature to quantify geometric agreement between the two methods; we consider the evidence convergent when cosine similarity exceeds 0.8.

Reproducibility. All experiments use PyTorch (Paszke et al. 2019) and `pyvene` (Wu et al. 2024).

5 Results

We present four groups of results: the succinctness separation (Section 5.1), DAS verification including negative controls (Section 5.2), SAE cross-validation (Section 5.3), and the DAS-IIA trajectory (Section 5.4). We conducted ablations over layer count, head count, weight decay, subspace dimension, and interpretability baselines, and extended the pipeline to n-bit binary decrement; key results are reported inline below.

Model	**#Params**	n=8	n=12	n=16	n=20	n=24
Transformer (ℓ=2)	403,840	$1.00_{\pm.00}$	$1.00_{\pm.00}$	$1.00_{\pm.00}$	$1.00_{\pm.00}$	$1.00_{\pm.00}$
LSTM	400,957	$.71_{\pm.08}$	$.28_{\pm.06}$	$.07_{\pm.03}$	$.02_{\pm.01}$	$.01_{\pm.01}$
GRU	402,483	$.65_{\pm.09}$	$.24_{\pm.07}$	$.05_{\pm.02}$	$.01_{\pm.01}$	$.00_{\pm.00}$
Mamba	403,674	$.68_{\pm.07}$	$.25_{\pm.05}$	$.06_{\pm.02}$	$.01_{\pm.01}$	$.00_{\pm.00}$

Table 1: Generalization results (test per-sequence accuracy, mean ± std over 5 seeds). The Transformer (ℓ=2) achieves perfect generalization via grokking at all n; all RNN and SSM baselines fail.

	B_{prev}	C_{in}	Y_{out}
(a) Component-level IIA			
L1-attn	$\mathbf{.98}_{\pm.01}$	$.53_{\pm.02}$	$.52_{\pm.02}$
L1-MLP	$.83_{\pm.03}$	$\mathbf{.96}_{\pm.02}$	$.59_{\pm.03}$
L2-attn	$.72_{\pm.04}$	$.69_{\pm.04}$	$.64_{\pm.04}$
L2-MLP	$.66_{\pm.05}$	$.92_{\pm.03}$	$\mathbf{.97}_{\pm.01}$
(b) Negative controls			
Y_{out}→AND, trained	$.98_{\pm.01}$	$.96_{\pm.02}$	$.52_{\pm.02}$
Y_{out}→OR, trained	$.98_{\pm.01}$	$.96_{\pm.02}$	$.54_{\pm.03}$
Correct, untrained	$.50_{\pm.02}$	$.51_{\pm.02}$	$.50_{\pm.02}$

Table 2: DAS verification at n=16 (mean ± std, 5 random seeds). **(a)** Component-level IIA; bold marks the best component c_i^* for each variable, confirming B-RASP predictions. **(b)** Negative controls; high IIA requires both a trained model and the correct SCM.

5.1 Generalization and Succinctness Separation

Transformer grokking. The Transformer (ℓ=2) achieves perfect test per-sequence accuracy (1.00) on all five bit widths after grokking. Table 1 reports the results. Memorization occurs within the first 600 to 4,200 steps depending on n, while generalization follows after a substantial delay: the grokking gap ranges from ∼7,200 steps at n=8 to ∼32,600 steps at n=24, with ∼16,800 steps at n=16 .

RNN and SSM baselines. All RNN and SSM models memorize the training set but fail to generalize. The degradation is exponential in n: the best RNN (LSTM) drops from 0.71 at n=8 to 0.07 at n=16 and 0.01 at n=24 (Table 1), a trajectory consistent with the $\Omega(2^n)$ lower bound on the required number of states. Mamba performs comparably to the GRU despite its selective scan mechanism, confirming that structured state space models do not circumvent the exponential barrier. Doubling RNN and SSM capacity to ∼800,000 parameters yields negligible improvement (best: LSTM 2× at 0.10 test per-sequence accuracy for n=16), providing further evidence that the separation is fundamental rather than an artifact of insufficient capacity.

5.2 DAS Verification of Causal Variables

Component-level IIA. At n=16, each variable achieves its highest IIA at the component predicted by the B-RASP specification: B_{prev} at L1-attn (0.98 ± 0.01), C_{in} at L1-MLP (0.96 ± 0.02), and Y_{out} at L2-MLP (0.97 ± 0.01). Table 2a reports the full component-level breakdown.

The structure of Table 2a reveals the sequential nature of the learned computation. C_{in} and Y_{out} show chance-level IIA (≈ 0.5) at L1-attn, confirming that neither variable is computed before the first MLP. C_{in} persists from L1-MLP through to L2-MLP (0.92) via the residual stream, consistent with the Transformer's residual connections making earlier computations accessible at later layers. This pattern matches the B-RASP dataflow: attention retrieves B_{prev} first, the first MLP computes C_{in} from the input bits, and the second MLP combines both to produce Y_{out}.

Variable	c_i^*	n=8	n=12	n=16	n=20	n=24
B_{prev}	L1-at	$.99_{\pm.01}$	$.99_{\pm.01}$	$.98_{\pm.01}$	$.98_{\pm.01}$	$.97_{\pm.02}$
C_{in}	L1-M	$.98_{\pm.01}$	$.97_{\pm.01}$	$.96_{\pm.02}$	$.96_{\pm.02}$	$.95_{\pm.02}$
Y_{out}	L2-M	$.99_{\pm.01}$	$.98_{\pm.01}$	$.97_{\pm.01}$	$.97_{\pm.02}$	$.96_{\pm.02}$

Table 3: DAS IIA at best component c_i^* across bit widths (mean ± std, 5 seeds). All variables maintain IIA > 0.95 for n=8 ∼ 24. Abbreviations: Ll-at $= (l, \texttt{attn})$, Ll-M $= (l, \texttt{mlp})$.

Variable	Component	Feature	$\lvert\rho\rvert$	Cls. Acc.	cos
B_{prev}	L1-attn	z_{37}	$.94_{\pm.02}$	$.97_{\pm.01}$	$.87_{\pm.03}$
C_{in}	L1-MLP	z_{142}	$.93_{\pm.02}$	$.96_{\pm.02}$	$.83_{\pm.04}$
Y_{out}	L2-MLP	z_{89}	$.96_{\pm.01}$	$.98_{\pm.01}$	$.91_{\pm.02}$

Table 4: SAE feature–variable alignment at n=16. Highest-correlating feature among D=1,024 is reported; all meet monosemanticity criteria ($|\rho| > 0.9$, classification accuracy ≥ 0.95). cos: cosine similarity between DAS direction and SAE decoder column.

Scaling across n. Table 3 reports IIA at c_i^* across all five bit widths. All three variables maintain IIA > 0.95 from n=8 to n=24. B_{prev} is nearly constant ($0.99 \rightarrow 0.97$), as expected for a position-independent attention copy. C_{in} shows the steepest decline ($0.98 \rightarrow 0.95$) because the carry chain $\bigwedge_{j=k+1}^{n} b_j$ grows with n; a per-position breakdown confirms this gradient. Even at n=24, the minimum IIA remains 0.95, demonstrating robust verification across the full range of carry chain lengths.

Subspace dimension ablation ($r \in \{1, 2, 4, 8\}$) and comparisons with alternative interpretability methods confirm that each Boolean variable is encoded in an approximately one-dimensional, non-axis-aligned subspace. Linear probes achieve high classification accuracy (≥ 0.97) for all variables, but when the probe direction is used for interchange interventions, IIA drops to 0.68–0.82; neuron-level patching yields still lower IIA (0.59–0.73). DAS's learned rotation consistently achieves the highest IIA, confirming that the causal variables are encoded in distributed, non-axis-

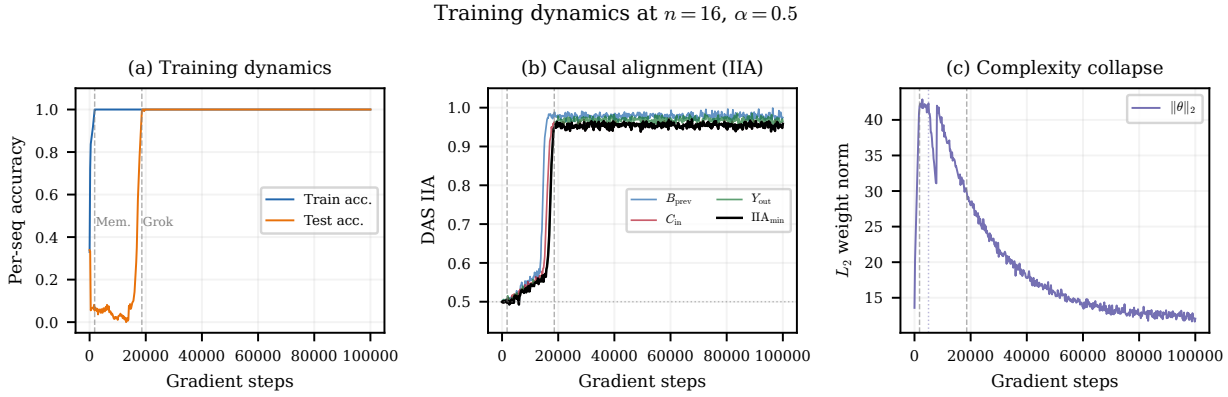


Figure 2: **Panel (a):** Per-sequence accuracy: the model memorizes by step ∼1,800 and generalizes at step ∼18,600. **(b):** DAS-IIA trajectory: individual variables and IIA_{min} (gray) transition from chance (0.5) to above 0.95 between steps ∼12,500 and ∼18,600. **(c):** L_2 weight norm: decline begins at step ∼5,000, well before the IIA transition. Dashed vertical lines mark the memorization and grokking steps.

aligned subspaces that require the rotation-based alignment. Notably, a single-layer Transformer (ℓ=1) achieves only $\text{IIA}_{\text{min}} = 0.61$ at n=16, and removing weight decay (λ=0) yields $\text{IIA}_{\text{min}} = 0.53$, confirming the B-RASP prediction that two processing stages and regularization-driven simplification are both necessary for specification-aligned structure to emerge .

Negative controls. Wrong-SCM variants (AND, OR replacing XOR in Y_{out}) yield chance-level IIA for Y_{out} (0.52–0.54), while the correct SCM on an untrained Transformer gives $\text{IIA}_{\text{min}} = 0.50$ (Table 2b). B_{prev} and C_{in} retain high IIA under the wrong-Y_{out} SCMs because only the Y_{out} equation is altered, confirming that each variable is verified independently. These controls establish *specificity*: DAS does not trivially find high-IIA rotations regardless of the hypothesis. The *validity* argument—that high IIA reflects actual causal structure rather than sensitivity to the output function—rests on a complementary line of evidence. DAS operates on intermediate hidden-layer activations (Eq. 5), not on the model's inputs or outputs; the probe–DAS dissociation above confirms this distinction: linear probes detect the *presence* of variable information (accuracy ≥ 0.97), but probe-direction interventions yield substantially lower IIA (0.68–0.82), demonstrating that information presence does not imply causal mediation. Cross-task controls further corroborate validity: applying the increment SCM to a decrement-trained model retains high IIA for the shared operation (B_{prev}: 0.97) but drops to chance for the task-specific variables (C_{in}: 0.54, Y_{out}: 0.52), a double dissociation confirming that the method discriminates at the level of individual structural equations, not merely at the level of input-output behavior.

5.3 SAE Cross-Validation

At each best component c_i^*, we trained an SAE and searched for monosemantic features aligning with the corresponding causal variable (Section 4.5). Table 4 summarizes the feature–variable alignment at n=16. For each of the three causal variables, the SAE discovers at least one feature satisfying both alignment criteria ($|\rho| > 0.9$ and classification accuracy ≥ 0.95). Each feature's activation pattern matches the corresponding B-RASP operation: bit copying for B_{prev}, conjunction for C_{in}, and exclusive disjunction for Y_{out} .

DAS–SAE geometric convergence. The rightmost column of Table 4 reports the cosine similarity between the DAS direction $R_{:,1}$ and the SAE decoder column of the best-matching feature. All three similarities exceed 0.83, meaning the two directions are nearly parallel in activation space, with Y_{out} achieving 0.91. This quantifies the convergent evidence: two methodologically independent analyses (supervised DAS with a causal objective, unsupervised SAE with a reconstruction objective) identify geometrically similar directions, substantially strengthening the conclusion that the Transformer implements the B-RASP specification. We emphasize that the SAE's role here is *geometric cross-validation* of the DAS-identified directions, not independent causal circuit discovery in the sense of (Marks et al. 2025); the causal evidence comes from DAS interventions, while the SAE confirms that the same structure is recoverable without supervision.

5.4 DAS-IIA Trajectory as Progress Measure

Figure 2 displays the training dynamics at n=16. We identify four phases based on the joint behavior of per-sequence accuracy (panel a) and specification alignment (panel b).

Phase 1: Random (0–1,800 steps). Both train and test per-sequence accuracy rise from their initial values but remain near chance. $\text{IIA}_{\text{min}} \approx 0.50$ for all three variables, indistinguishable from the untrained-model control (Table 2).

Phase 2: Memorization (1,800–12,500 steps). IIA_{min} rises slowly toward 0.56, far below the 0.95 threshold. Notably, B_{prev} exhibits a transient "false start" between steps 6,000 and 12,000, briefly reaching ∼0.58, suggesting that the attention copy pattern forms intermittently before stabilizing. This false start does not propagate to C_{in} or Y_{out}, indicating that the full causal chain has not yet assembled.

Phase 3: Phase transition (12,500–18,600 steps). The variables align sequentially, i.e., B_{prev} first (centered at step $\sim$14,500), then C_{in} ($\sim$16,000), then Y_{out} ($\sim$17,200), mirroring the causal dependency chain: the output computation cannot align until its inputs are in place. Test per-sequence accuracy rises in tandem, confirming that generalization and specification alignment are tightly coupled.

Phase 4: Convergence (18,600+ steps). $\text{IIA}_{\min}$ stabilizes at 0.955 ± 0.006; no further structural reorganization occurs.

Comparison with L_2 weight norm. The L_2 weight norm (panel c) begins declining $\sim$7,500 steps before the IIA transition and does so gradually, lacking any sharp change. This contrast highlights the advantage of a specification-aligned measure: weight norm signals that *something* is simplifying, whereas $\text{IIA}_{\min}$ reveals when the specific logical structure prescribed by the B-RASP specification is acquired. This trajectory analysis is an application of the verification framework (Section 3.3), not a separate contribution: it demonstrates that the same KR specification used for static verification can also serve as a progress measure during training, illustrating the broader diagnostic utility of KR formalisms.

6 Conclusion and Future Work

Formal language theory tells us what Transformers *can* represent; this paper asks whether gradient-trained Transformers *do* represent what the theory predicts. By compiling B-RASP specifications into testable causal models, we turn this from a philosophical question into an empirical one with clear pass/fail criteria. The answer, for n-bit binary increment, is affirmative: not only does the trained Transformer implement the prescribed logical circuit, but it acquires this structure through a sharp phase transition that existing task-agnostic progress measures cannot detect.

This finding has a broader implication for the KR community. KR formalisms are typically valued for their *prescriptive* role: specifying what a system should compute. Our results show they can equally serve a *diagnostic* role: as falsifiable causal hypotheses tested against trained networks via mechanistic interpretability. The novelty lies not in the individual ML techniques but in the closed-loop bridge from formal specification to mechanistic neural verification. Causal abstraction enables this bridge bidirectionally: KR specifications constrain which hypotheses to test, while empirical outcomes feed back into the formalism.

Several limitations suggest directions for future work: extending to structurally richer B-RASP programs, relaxing the $r{=}1$ subspace assumption for tasks with richer intermediate representations, and automating B-RASP-to-SCM compilation for broader specification classes. A complementary direction is to apply the framework to pre-trained language models on tasks where a B-RASP specification exists; DAS has already been deployed at this scale, so the verification method transfers, though whether the clean one-variable-one-direction structure observed here persists in larger, multi-task networks remains an open question.

Acknowledgments

We would like to thank the reviewers for their useful comments and constructive suggestions. This work was partially supported by the National Natural Science Foundation of China (No. 62476125), the Noncommunicable Chronic Diseases–National Science and Technology Major Project (No. 2025ZD0550704), the Fundamental Research Funds for the Central Universities (No. 0221-14380025), the "111 Center" (No. B26023), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118), and the Nanjing University International Collaboration Initiative.

Use of Generative AI

During the preparation of this paper, the authors used generative AI tools to assist with copyediting, grammar checking, and improving clarity and readability. All technical claims, experimental designs, numerical results, references, and conclusions were developed, checked, and finalized by the authors. The authors take full responsibility for the accuracy, originality, integrity, and final form of this paper.

References

Baral, C., and Giacomo, G. D. 2015. Knowledge Representation and Reasoning: What's Hot. In *Proc. AAAI'15*, 4316–4317. AAAI Press.

Bergsträßer, P.; Cotterell, R.; and Lin, A. W. 2026. Transformers are Inherently Succinct. In *Proc. ICLR'26*. OpenReview.net.

Bhattamishra, S.; Patel, A.; Kanade, V.; and Blunsom, P. 2023. Simplicity Bias in Transformers and their Ability to Learn Sparse Boolean Functions. In *Proc. ACL'23*, 5767–5791. ACL.

Bhattamishra, S.; Hahn, M.; Blunsom, P.; and Kanade, V. 2024. Separations in the Representational Capabilities of Transformers and Recurrent Architectures. In *Advances in NeurIPS'24*.

Carvalho, B. W.; d'Avila Garcez, A. S.; Lamb, L. C.; and Brazil, E. V. 2025. Grokking Explained: A Statistical Phenomenon. *CoRR* abs/2502.01774.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. EMNLP'14*, 1724–1734. ACL.

Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; DasSarma, N.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Elhage, N.; Hume, T.; Olsson, C.; Schiefer, N.; Henighan, T.; Kravec, S.; Hatfield-Dodds, Z.; Lasenby, R.; Drain, D.; Chen, C.; Grosse, R. B.; McCandlish, S.; Kaplan, J.; Amodei, D.; Wattenberg, M.; and Olah, C. 2022. Toy Models of Superposition. *CoRR* abs/2209.10652.

Elman, J. L. 1990. Finding structure in time. *Cogn. Sci.* 14(2):179–211.

Geiger, A.; Lu, H.; Icard, T.; and Potts, C. 2021. Causal Abstractions of Neural Networks. In *Advances in NeurIPS'21*, 9574–9586.

Geiger, A.; Wu, Z.; Potts, C.; Icard, T.; and Goodman, N. D. 2024. Finding Alignments Between Interpretable Causal Variables and Distributed Neural Representations. In *Proc. CLeaR'24*, volume 236 of *Proceedings of Machine Learning Research*, 160–187. PMLR.

Geirhos, R.; Jacobsen, J.; Michaelis, C.; Zemel, R. S.; Brendel, W.; Bethge, M.; and Wichmann, F. A. 2020. Shortcut learning in deep neural networks. *Nat. Mach. Intell.* 2(11):665–673.

Gu, A., and Dao, T. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *CoRR* abs/2312.00752.

Gu, A.; Goel, K.; and Ré, C. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *Proc. ICLR'22*. OpenReview.net.

Hahn, M. 2020. Theoretical Limitations of Self-Attention in Neural Sequence Models. *Trans. Assoc. Comput. Linguistics* 8:156–171.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. ICCV'15*, 1026–1034. IEEE Computer Society.

Heimersheim, S., and Nanda, N. 2024. How to use and interpret activation patching. *CoRR* abs/2404.15255.

Hendrycks, D., and Gimpel, K. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR* abs/1606.08415.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.* 9(8):1735–1780.

Huben, R.; Cunningham, H.; Smith, L. R.; Ewart, A.; and Sharkey, L. 2024. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *Proc. ICLR'24*. OpenReview.net.

Jain, S., and Wallace, B. C. 2019. Attention is not Explanation. In *Proc. NAACL-HLT'19*, 3543–3556. ACL.

Kumar, T.; Bordelon, B.; Gershman, S. J.; and Pehlevan, C. 2024. Grokking as the Transition from Lazy to Rich Training Dynamics. In *Proc. ICLR'24*. OpenReview.net.

Lieto, A. 2021. *Cognitive Design for Artificial Minds*. Routledge.

Lindner, D.; Kramár, J.; Farquhar, S.; Rahtz, M.; McGrath, T.; and Mikulik, V. 2023. Tracr: Compiled Transformers as a Laboratory for Interpretability. In *Advances in NeurIPS'23*.

Loshchilov, I., and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *Proc. ICLR'19*. OpenReview.net.

Lyu, K.; Jin, J.; Li, Z.; Du, S. S.; Lee, J. D.; and Hu, W. 2024. Dichotomy of Early and Late Phase Implicit Biases Can Provably Induce Grokking. In *Proc. ICLR'24*. OpenReview.net.

Makelov, A.; Lange, G.; Geiger, A.; and Nanda, N. 2024. Is This the Subspace You Are Looking for? An Interpretability Illusion for Subspace Activation Patching. In *Proc. ICLR'24*. OpenReview.net.

Marks, S.; Rager, C.; Michaud, E. J.; Belinkov, Y.; Bau, D.; and Mueller, A. 2025. Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models. In *Proc. ICLR'25*. OpenReview.net.

McNaughton, R., and Papert, S. 1971. *Counter-Free Automata*. Cambridge, MA: MIT Press.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and Editing Factual Associations in GPT. In *Advances in NeurIPS'22*.

Merrill, W., and Sabharwal, A. 2024. The Expressive Power of Transformers with Chain of Thought. In *Proc. ICLR'24*. OpenReview.net.

Merrill, W.; Weiss, G.; Goldberg, Y.; Schwartz, R.; Smith, N. A.; and Yahav, E. 2020. A Formal Hierarchy of RNN Architectures. In *Proc. ACL'20*, 443–459. ACL.

Mueller, A.; Geiger, A.; Wiegreffe, S.; Arad, D.; Arcuschin, I.; Belfki, A.; Chan, Y. S.; Fiotto-Kaufman, J. F.; Haklay, T.; Hanna, M.; Huang, J.; Gupta, R.; Nikankin, Y.; Orgad, H.; Prakash, N.; Reusch, A.; Sankaranarayanan, A.; Shao, S.; Stolfo, A.; Tutek, M.; Zur, A.; Bau, D.; and Belinkov, Y. 2025. MIB: A Mechanistic Interpretability Benchmark. In *Proc. ICML'25*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.

Nanda, N.; Chan, L.; Lieberum, T.; Smith, J.; and Steinhardt, J. 2023. Progress measures for grokking via mechanistic interpretability. In *Proc. ICLR'23*. OpenReview.net.

Nerode, A. 1958. Linear Automaton Transformations. *Proceedings of the American Mathematical Society* 9(4):541–544.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom In: An Introduction to Circuits. *Distill*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in NeurIPS'19*, 8024–8035.

Pearl, J. 2009. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. FOCS'77*, 46–57. IEEE Computer Society.

Power, A.; Burda, Y.; Edwards, H.; Babuschkin, I.; and Misra, V. 2022. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. *CoRR* abs/2201.02177.

Sanford, C.; Hsu, D. J.; and Telgarsky, M. 2023. Representational Strengths and Limitations of Transformers. In *Advances in NeurIPS'23*.

Shah, H.; Tamuly, K.; Raghunathan, A.; Jain, P.; and Netrapalli, P. 2020. The pitfalls of simplicity bias in neural networks. In *Advances in NeurIPS'20*.

Sharkey, L.; Chughtai, B.; Batson, J.; Lindsey, J.; Wu, J.; Bushnaq, L.; Goldowsky-Dill, N.; Heimersheim, S.; Ortega, A.; Bloom, J. I.; Biderman, S.; Garriga-Alonso, A.; Conmy, A.; Nanda, N.; Rumbelow, J.; Wattenberg, M.; Schoots, N.; Miller, J.; Saunders, W.; Michaud, E. J.; Casper, S.; Tegmark, M.; Bau, D.; Todd, E.; Geiger, A.; Geva, M.; Hoogland, J.; Murfet, D.; and McGrath, T. 2025. Open Problems in Mechanistic Interpretability. *Trans. Mach. Learn. Res.* 2025.

Strobl, L.; Merrill, W.; Weiss, G.; Chiang, D.; and Angluin, D. 2024. What formal languages can transformers express? a survey. *Trans. Assoc. Comput. Linguistics* 12:543–561.

Strobl, L.; Angluin, D.; Chiang, D.; Rawski, J.; and Sabharwal, A. 2025. Transformers as Transducers. *Trans. Assoc. Comput. Linguistics* 13:200–219.

Sun, J.; Huang, J.; Baskaran, S.; D'Oosterlinck, K.; Potts, C.; Sklar, M.; and Geiger, A. 2025. HyperDAS: Towards Automating Mechanistic Interpretability with Hypernetworks. In *Proc. ICLR'25*. OpenReview.net.

Templeton, A.; Conerly, T.; Marcus, J.; Lindsey, J.; Bricken, T.; Chen, B.; Pearce, A.; Citro, C.; Ameisen, E.; Jones, A.; Cunningham, H.; Turner, N. L.; McDougall, C.; MacDiarmid, M.; Tamkin, A.; Durmus, E.; Hume, T.; Mosconi, F.; Freeman, C. D.; Sumers, T. R.; Rees, E.; Batson, J.; Jermyn, A.; Carter, S.; Olah, C.; and Henighan, T. 2024. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in NeurIPS'17*, 5998–6008.

Wang, K. R.; Variengien, A.; Conmy, A.; Shlegeris, B.; and Steinhardt, J. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *Proc. ICLR'23*. OpenReview.net.

Weiss, G.; Goldberg, Y.; and Yahav, E. 2021. Thinking Like Transformers. In *Proc. ICML'21*, volume 139 of *Proceedings of Machine Learning Research*, 11080–11090. PMLR.

Wiegreffe, S., and Pinter, Y. 2019. Attention is not not Explanation. In *Proc. EMNLP-IJCNLP'19*, 11–20. ACL.

Wu, Z.; Geiger, A.; Icard, T.; Potts, C.; and Goodman, N. D. 2023. Interpretability at Scale: Identifying Causal Mechanisms in Alpaca. In *Advances in NeurIPS'23*.

Wu, Z.; Geiger, A.; Arora, A.; Huang, J.; Wang, Z.; Goodman, N. D.; Manning, C. D.; and Potts, C. 2024. pyvene: A Library for Understanding and Improving PyTorch Models via Interventions. In *Proc. NAACL'24*, 158–165. ACL.

Wu, Y.; Geiger, A.; and Millière, R. 2025. How Do Transformers Learn Variable Binding in Symbolic Programs? In *Proc. ICML'25*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net.

Yang, A.; Chiang, D.; and Angluin, D. 2024. Masked Hard-Attention Transformers Recognize Exactly the Star-Free Languages. In *Advances in NeurIPS'24*.