VERITAS: VERified Interpretations of Transformer Activations

Anonymous authors
Paper under double-blind review

Abstract

Activation interpretability faces a fundamental credibility gap: a learned reader can reconstruct a hidden state through a bottleneck that is semantically wrong. Reconstruction constrains information flow; it does not fix meaning. We introduce Veritas, which closes this gap by replacing self-authored semantics with executable semantics. Before any activation is read, an external executor runs a finite set of candidate actions, normalizes their observed successor states, and exposes the resulting codebook to the reader. The reader may select a candidate identity but cannot redefine what that identity denotes—denotation is owned by the verifier, not the interpreter.

We instantiate Veritas in Lean theorem proving with Goedel-Prover-V2-8B across 1,666 proof transitions. Four complementary measurements establish the interface's properties. Externally fixed candidate identity improves held-out activation prediction by $\Delta\text{FVE}_{\text{id}} = 0.0454$ beyond a decoder that already sees the full candidate set (95% CI [0.0291, 0.0623]). An activation-conditioned pointer reaches 80.9% candidate-preference accuracy versus 73.6% for the strongest non-activation baseline. Candidate-conditioned reconstructions steer the frozen model toward their target candidate with a 0.114 argmax-match advantage over matched arbitrary-code reconstructions (95% CI [0.077, 0.158]). And while state-aware attackers readily recover an injected nuisance signal from an equal-cardinality free code, the grounded pointer yields near-zero incremental leakage across a 30× attacker-capacity range. The core reconstruction and decoding signals replicate in the Qwen3 parent model under the same candidate environment.

Veritas points to a broader design principle: let an executable system own the semantics of the latent channel, then probe how neural states predict, reconstruct through, and steer via it. Lean is the first testbed; the same interface extends naturally to sandboxes, unit tests, simulators, and typed tool effects—turning activation interpretability into an actionable foundation for white-box AI control.

1 Introduction

Mechanistic interpretability is becoming increasingly capable of extracting structured information from neural activations, yet the semantics of that information are usually supplied by the interpreter itself. Learned readers, autoencoders, and language decoders can tell us what information is recoverable from a hidden state, but a successful readout does not by itself determine what the intermediate code means. In parallel, formal methods are beginning to enter interpretability, using verification to certify properties of circuits and replacement models (Hadad et al., 2026; Ladner & Althoff, 2026). These two lines of work address complementary parts of the same trust problem: activation readers expose internal structure; formal systems make specified semantics checkable. Veritas combines them by using an executable verifier to define the interpreter's semantic alphabet before the activation is read.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

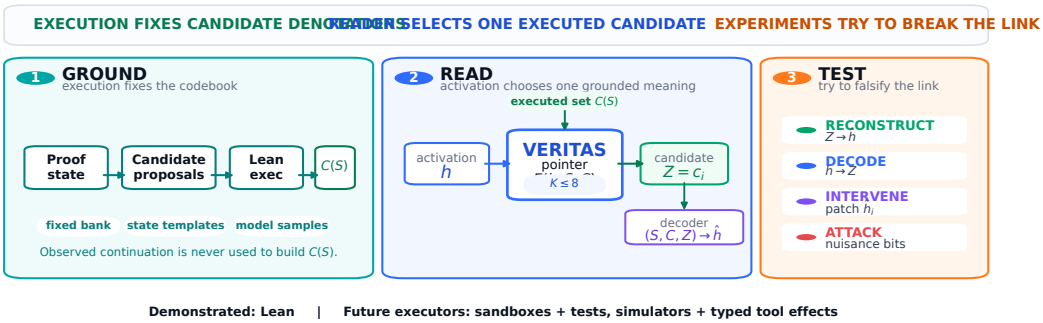

Figure 1: VERITAS fixes candidate denotations outside the activation reader. Candidate actions are proposed without inspecting the realized continuation, executed from the same Lean state, then normalized and deduplicated by observed successor state to form $C(S)$. Conditional on the shared public context (S, C), the reader may only select an identity $Z \in C(S)$; a decoder may reconstruct $\hat{h}$ from (S, C, Z). Prediction, reconstruction, intervention, and nuisance-leakage tests then probe how neural state relates to this externally grounded codebook.

Natural Language Autoencoders (NLAs) make the semantic-contract problem especially clear. NLAs map activations through natural language and back (Fraser-Taliente et al., 2026), but reconstruction can remain nearly unchanged after initializing the language bottleneck with semantically implausible explanations (Zhang & Turner, 2026). The failure is more general than language: any sufficiently expressive learned bottleneck can preserve information in degrees of freedom that are not fixed by its apparent interpretation. Reconstruction constrains information flow; it does not fix denotation.

VERITAS (VERified Interpretations of Transformer Activations) replaces self-authored semantics with executable semantics. For each state S, an external executor constructs a finite set $C(S)$ of counterfactual actions, executes each action, and assigns a deterministically normalized identity to its observed successor. The activation reader receives h and may only select an identity $Z \in C(S)$. It cannot invent a new symbol, rewrite a candidate, or change what the selected identity denotes. In Lean, this contract is unusually clean: candidate tactics are run from the same proof state, invalid actions are rejected, and legal successors are normalized before the reader is trained or evaluated (Figure 1). The central move is simple: reconstruction tells us that a code carries information; the verifier fixes what the code is allowed to denote.

This change in contract makes faithfulness experimentally testable. We ask whether candidate identity adds predictive value beyond the public state and candidate set; whether the activation identifies the model's scoring-defined preferred candidate beyond strong lexical baselines; whether candidate-conditioned reconstructions cause target-specific behavioral shifts; and whether the grounded channel resists an injected nuisance signal that an equal-cardinality free code can transmit. We also replay the locked evaluation on the Qwen3 parent. Across these tests, candidate identity adds 0.0454 held-out FVE beyond the candidate-set decoder, the activation-conditioned pointer reaches 80.9% accuracy versus 73.6% for the strongest observed no-activation baseline, candidate-conditioned reconstructions produce a theorem-bootstrap steering advantage of 0.114 over arbitrary-code reconstructions, and state-aware attackers readily decode the nuisance from the free code while finding almost no incremental signal in the grounded pointer.

The broader opportunity is to turn interpretability outputs into executable interfaces. Lean is a deliberately crisp domain in which candidate effects are cheap to execute and easy to check. The same abstraction extends wherever counterfactual actions can be evaluated by an external system: sandboxes and tests for coding agents, transaction simulators for databases, typed effect logs for tools, and simulators for embodied systems. This suggests

a future white-box architecture in which internal activations are mapped into a small set of externally checkable action effects that can be monitored, deferred, or directly intervened on. This paper establishes the core interface and demonstrates the key ingredients needed for that extension.

2 Related Work

From activation readers to semantic interfaces. A rapidly growing class of methods learns human-readable or behaviorally useful views of neural activations. NLAs map activations to generated text and back (Fraser-Taliente et al., 2026); Activation Oracles and follow-up work use learned language-model readers and document hallucination, inversion, and concept-specific blind spots (Karvonen et al., 2025; Bauer et al., 2026; Bersia & Gaintseva, 2026); Predictive Concept Decoders route activations through a sparse concept bottleneck before predicting model behavior (Huang et al., 2025); and verbalizable-representation work asks when neural states admit compact linguistic readouts (Gurnee et al., 2026). These systems differ in architecture, but they share a key design choice: the semantic vocabulary is learned or generated by the interpreter. Veritas changes that choice. Its vocabulary is fixed by externally executed transitions, so the reader selects among pre-grounded meanings rather than jointly learning a code and its interpretation.

Concept bottlenecks, causal abstraction, and formal verification. Concept Bottleneck Models introduced supervised human concepts as an explicit intermediate representation and showed that concept-level intervention can make predictions inspectable and editable (Koh et al., 2020); later LLM, verified-neuron, programmatic-bottleneck, and neurosymbolic work extends structured bottlenecks to richer domains (Sun et al., 2024; Ji et al., 2026; Duan et al., 2026; Dhanraj & Eliasmith, 2025). Causal abstraction provides a rigorous criterion for when a high-level variable faithfully captures neural computation, and Distributed Alignment Search searches for such variables in distributed representations (Geiger et al., 2025; 2024). Veritas is complementary: it fixes the denotation of the candidate variable externally, then measures predictive correspondence and target-specific steerability. Recent work has also begun to bring formal verification directly into mechanistic interpretability, certifying robustness, patching, minimality, or reachability properties of discovered circuits and interpretable replacement networks (Hadad et al., 2026; Ladner & Althoff, 2026). VERITAS uses formal execution at a different boundary: not to certify a discovered circuit, but to make the semantic interface itself externally checkable.

Formal reasoning, intervention, and AI control. Theorem proving provides a natural first domain because Lean exposes executable proof states and machine-checked transitions. We build on Goedel-Prover-V2 and LeanDojo (Lin et al., 2026; Yang et al., 2023); activation steering in theorem provers separately shows that residual-stream interventions can change proof behavior (Kirtania, 2025). Beyond theorem proving, AI control studies protocols that remain useful when powerful models may act adversarially (Greenblatt et al., 2024; Kutasov et al., 2025), and white-box control explicitly considers monitoring internal activations (AI Security Institute, 2025). ControlArena provides adversarial environments for evaluating such protocols (Inglis et al., 2025), while narrow information channels have been proposed as a way to bound influence from untrusted models (Biddulph & Kaufman, 2026). VERITAS connects these directions by pairing a bounded latent choice with externally executed denotations, creating an interface that is both interpretable and directly usable for intervention.

3 Method: A Verifier-Grounded Semantic Interface

3.1 The interface

Let S denote an externally observable task state and let an executor produce a finite candidate set

$$C(S) = \{c_1, \ldots, c_K\}, \tag{1}$$

where each c_i denotes a Lean-executed action together with a deterministically normalized successor-state identity. We cap K at 8 in the experiments below. For a hidden activation $\mathbf{h}$ extracted at the same decision point, a Veritas encoder is a pointer

$$E(\mathbf{h}, S, C) \to Z \in \{1, \ldots, K\}, \tag{2}$$

and the reconstruction decoder maps (S, C, Z) back to $\hat{\mathbf{h}}$. Conditional on the shared public context (S, C), Z is the only bottleneck variable. The structural constraint is the key idea: the encoder may choose among externally fixed candidate denotations but cannot modify their content.

Legality alone is not enough. Arbitrary true statements, aliases, ordering choices, or equivalent actions could still form covert channels. We therefore normalize successor-state strings, deterministically deduplicate them, and truncate candidates by a fixed ordering. This is conservative normalization, not theorem-proving equivalence: logically equivalent transitions can remain distinct. Verification constrains interpreter-controlled degrees of freedom in the codebook; it does not prove faithfulness or eliminate all possible covert channels.

3.2 Lean instantiation and target semantics

The frozen target is Goedel-Prover-V2-8B (Lin et al., 2026), a 36-layer Qwen3-family theorem prover with hidden width 4096; we cite the Qwen3 base family separately (Yang et al., 2025). We capture the layer-30 residual activation at the last model token before each Lean tactic boundary using deterministic teacher-forced replay of a Lean-accepted proof. Layer 30 was selected using a validation-only sweep on 12 development theorems (Appendix A).

For each proof state, candidates come from three sources that never inspect the realized continuation: a fixed tactic bank, state-derived templates, and independent frozen-model samples from the exact pre-tactic prefix. Because one proposal source is Goedel itself, $C(S)$ is not model-neutral: it is independent of the realized continuation and learned activation reader, but it is partly shaped by Goedel's proposal distribution. Every proposal is executed from the same Lean state. Invalid proposals are removed; legal successors are normalized, deduplicated, and deterministically truncated to $K \leq 8$. Only after $C(S)$ is fixed do we check whether it contains the sampled trajectory transition. This rule corrects a label-leakage bug in our first implementation, which inserted the observed tactic before truncation and produced a spurious 95% accuracy (Appendix C).

The sampled rollout is not necessarily the candidate favored by the frozen model under a fixed scoring rule. We define the primary target as

$$Z^*(S) = \arg \max_{c_i \in C(S)} \log p_\theta(c_i \mid \text{prefix}), \tag{3}$$

where $\log p_\theta(c_i \mid \text{prefix})$ is the unnormalized total teacher-forced log-probability of the candidate tactic string. This rule is length-sensitive. Accordingly, "preferred" throughout means preferred under this fixed candidate-scoring rule, not necessarily the action the model would choose under unconstrained generation. Sampled trajectory and scoring-defined preferred candidate disagree on 42.5% of retained states, so the distinction materially changes the label.

3.3 Evaluation: explain, decode, intervene, attack

We use three matched ridge decoders to separate public context from candidate identity:

$$D_0(S) \to \hat{h}, \qquad D_C(S, C) \to \hat{h}, \qquad D_1(S, C, Z^*) \to \hat{h}. \tag{4}$$

State and candidate text are represented with fixed hashed n-gram features. Activations are unit-ℓ_2 normalized for reconstruction, and we report fraction of variance explained as $\text{FVE} = 1 - \sum_i \|h_i - \hat{h}_i\|_2^2 / \sum_i \|h_i - \bar{h}\|_2^2$ using a fixed global-mean reference. The load-bearing reconstruction statistic is

$$\Delta\text{FVE}_{\text{id}} = \text{FVE}(D_1) - \text{FVE}(D_C), \tag{5}$$

not the larger $D_1 - D_0$ difference, because D_C controls for the information available to this featurized candidate-set decoder. $\Delta\text{FVE}_{\text{id}}$ is therefore a decoder- and featurization-dependent incremental prediction statistic, not a decoder-independent mutual-information estimate.

For candidate decoding we compare the activation-conditioned pointer against non-activation baselines, including explicit first-token and action-family features. We also use shuffled activations, candidate-order permutations, zero and fixed-noise activation branches, and a retrained-permutation null with 200 fresh models per theorem-level CV fold. Reconstruction uses six-fold theorem-level cross-validation and theorem-level bootstrap intervals (2,000 draws). For causal evaluation, we reconstruct one activation per candidate, patch it at the original layer/token, and rescore all candidates. This tests candidate-conditioned steering, not whether Z is a proved causal abstraction of the original network. For nuisance-leakage evaluation, we inject independent bits and train state-aware attackers that observe (S, C, Z).

The equal-cardinality arbitrary-code baseline keeps the same per-state K and matched encoder/decoder capacity but replaces grounded candidate features with unconstrained learned slot embeddings. It therefore asks what a free K-way code can carry at the same nominal cardinality. Because the grounded pointer and free code differ in both semantics and training constraints, this is not by itself a pure isolation of "verification" as a causal factor. All train/test splits are by theorem, and uncertainty is bootstrapped at the theorem level where reported.

4 Experiments and Results

Figure 2 summarizes the central evidence: prediction, decoding, steering, and matched-parent measurements probe different aspects of the proposed interface.

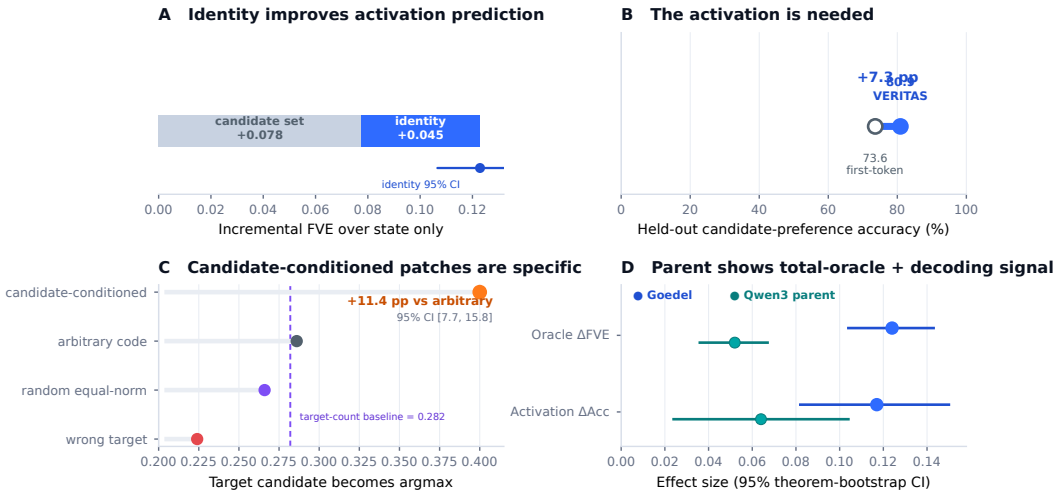

Figure 2: Four complementary measurements. (A) Candidate identity improves activation prediction beyond the paper's state-and-candidate featurization. (B) Activation-conditioned decoding has a 7.3-point higher held-out point estimate than the strongest observed no-activation baseline; no theorem-clustered interval is available for this specific contrast. (C) Candidate-conditioned reconstructions steer the corresponding target relative to arbitrary-code and wrong-target patches. (D) In the shared Goedel-constructed candidate sets, Qwen3 shows a positive total oracle reconstruction gate and activation-decoding effect; the identity-only reconstruction decomposition was not rerun for Qwen. Error bars are 95% theorem-bootstrap intervals where available.

4.1 Q1: candidate identity adds predictive value and can be decoded from the activation

We generated two sampled proof attempts for each of the 244 theorems in the MiniF2F test split (488 rollouts total) (Zheng et al., 2022). Only Lean-accepted generated proofs enter tactic-boundary replay, so the analysis is additionally conditioned on successful proof generation. The current pipeline yields a selected set of 1,666 retained transitions across 109 test theorems. Retention still requires the sampled trajectory successor to be present even though the primary preferred-candidate target does not require trajectory coverage; this avoidable selection rule is analyzed in Appendix B. Candidate construction covers 63.1% of attempted tactic boundaries, with dominant failures from compound have blocks.

Table 1: Candidate-identity prediction and decoding. D_C controls for the available candidate set under the paper's hashed feature representation. The first-token classifier is the strongest observed no-activation baseline.

Test	Estimate	95% CI / control
Total $D_1 - D_0$ ΔFVE	0.1228	[0.1040, 0.1429]
Candidate-set contribution $D_C - D_0$	0.0775	[0.0701, 0.0854]
Identity contribution $D_1 - D_C$	0.0454	[0.0291, 0.0623]
VERITAS pointer accuracy	80.9%	theorem-CV
Strong context + lexical features	72.7%	no activation
First-token-only classifier	73.6%	strongest no-activation
Conservative activation margin	+7.3 pp	descriptive point estimate

Identity improves activation prediction beyond the state and available choices. Candidate identity contributes 0.0454 FVE beyond (S, C), with a theorem-bootstrap interval entirely above zero (Table 1). This is an incremental-prediction result, not a claim of high-fidelity autoencoding: the absolute ridge-decoder FVE is near or below zero in most folds, so the result should be read as candidate identity improving a weak decoder rather than reconstructing the activation well in absolute terms. In the reverse direction, Veritas reaches 80.9% held-out accuracy. A simple first-token classifier reaches 73.6%, revealing that much of this late-layer task is lexically predictable. We therefore report the 7.3-point strongest-baseline gap as a descriptive point estimate; unlike the formal-versus-context contrast, this specific difference does not have a theorem-clustered confidence interval. This target is deliberately close to model output: layer 30 is six transformer blocks before the logits, and Z^* is defined from the same model's continuation scores. The result therefore establishes a late-layer representation of candidate preference under our scoring rule, not an abstract semantic variable independent of action prediction. Stronger same-family and same-prefix candidate sets, alternative length-normalized scoring rules, and logit-level baselines are important future controls.

The effect survives controls aimed at architecture and label structure. None of 1,200 retrained-permutation models matches the corresponding real-activation model ($p = 0.005$, the minimum possible with 200 permutations, in all six folds). Zero and fixed-noise activation branches track no-activation performance, and candidate-order permutation is exactly equivariant after remapping labels. The supported conclusion is therefore narrower than a mutual-information claim: under the specified featurization and decoder family, the activation carries incremental predictive signal about which executed candidate the frozen model scores highest.

4.2 Q2: candidate-conditioned reconstructions produce target-specific steering

Decodability alone does not establish causal relevance. For each held-out state and candidate c_i, we reconstruct $\hat{h}_i = D(S, C, Z = i)$, patch it at the original layer and token, and rescore all candidates. We compare this candidate-conditioned patch against state-only reconstruction, zero, random equal-norm, wrong-target candidate reconstruction, and a matched arbitrary-code reconstruction. This intervention asks whether a learned candidate-

conditioned prototype can steer model preference; it is weaker than showing that candidate identity is a natural causal mediator in the original computation.

Table 2: Controlled patching on 238 held-out records and 843 target-scoring events. Because target-independent patches are evaluated once against every candidate target, their aggregate argmax-match rate is mechanically $1/K$ per record; the informative comparisons are the target-conditioned semantic, arbitrary-code, random, and wrong-target patches.

Patch	Argmax match	vs. $1/K$	Mean rank shift
Verified candidate reconstruction	0.400	+0.118	+0.206
State-only (target-independent)	0.282	0.000	0.000
Zero (target-independent)	0.282	0.000	0.000
Random equal-norm	0.266	-0.016	+0.009
Wrong verified target	0.224	-0.058	-0.079
Arbitrary-code reconstruction	0.286	+0.004	-0.015

Candidate-conditioned reconstruction produces target-specific steering. The informative result is the separation among target-conditioned patches. The candidate-conditioned reconstruction reaches 0.400 argmax match versus 0.286 for the matched arbitrary-code reconstruction, with theorem-bootstrap $\Delta_{\text{causal}} = 0.114$ and 95% CI [0.077, 0.158]. Random equal-norm patches remain near the target-count baseline, while patching a different verified candidate pushes the evaluated target below it (0.224), supporting directional specificity. The state-only and zero rows are target-independent calibration checks: because each such patch is scored once against every candidate target in a record, their aggregate argmax-match rate is mechanically $1/K$ and should not be read as independent evidence. The decoder was trained only for reconstruction, so the target-specific steering effect was not optimized directly. It nevertheless may learn candidate-conditioned activation prototypes; these results establish steerability, not that Z is a proved causal abstraction of the original model. Interchange interventions or necessity tests on naturally occurring representations would support the stronger claim.

4.3 Q3: the grounded pointer shows low leakage on the tested nuisance channel

An equal-cardinality free code is the primary structural control. The arbitrary-code baseline has the same per-state cardinality K and matched encoder/decoder capacity but no executed candidate denotations. Under the calibrated nuisance injection, the free code readily carries the unrelated bits while the grounded pointer does not under the tested attackers (below). This is the most direct capacity-matched evidence in Q3. Even here, we avoid attributing the gap to "verification alone": the grounded and free channels differ in both semantic constraint and training target.

State-aware attackers recover the injected nuisance from the free code, with near-zero incremental accuracy for the grounded pointer. Because the denotation of a pointer depends on (S, C), marginal $I(Z; B)$ can miss context-dependent encodings. We therefore compare a blind attacker $(S, C) \to B$ with a state-aware attacker $(S, C, Z) \to B$. Across linear, small-MLP, and large-MLP attackers spanning more than 30-fold parameter count, the incremental state-aware accuracy of Veritas remains only 0.002 to 0.008, while the equal-cardinality arbitrary code remains 0.25 to 0.28. At a nuisance-injection operating point calibrated using the arbitrary code alone, the sum of per-bit marginal mutual-information estimates is 0.044 bits for Veritas versus 0.920 bits for the arbitrary code across five seeds. The 0.002–0.008 and 0.25–0.28 ranges above are ranges across attacker capacities, not confidence intervals. These synthetic tests do not establish a general covert-capacity bound: they use one injected orthonormal nuisance construction, one calibrated operating point, and supervised attackers. They show that the free code readily encodes this nuisance while the grounded pointer does not under the tested attacks.

4.4 Q4: reconstruction and decoding signals are positive in the matched parent

Goedel-Prover-V2-8B is derived from Qwen3-8B (Yang et al., 2025), enabling a matched-parent replay on the same 1,666 prefixes, theorem folds, layer index, tokenizer, and candidate sets. The candidate sets are not model-neutral: they were constructed by the Goedel pipeline and include Goedel-sampled proposals. Qwen is therefore evaluated in a shared Goedel-constructed candidate environment. For Qwen, candidate scores, preferred labels, and top1-to-top2 margins are recomputed from Qwen's own teacher-forced probabilities over that fixed set.

Table 3: Shared-candidate matched-parent comparison. Stage C repeats the original total oracle gate $D_1 - D_0$ and activation ΔAcc, not the later identity-only $D_1 - D_C$ decomposition. The Goedel-constructed candidate sets are held fixed while each model is rescored on them.

Metric	Goedel-Prover-V2-8B	Qwen3-8B
Oracle ΔFVE	0.124 [0.104, 0.143]	0.052 [0.036, 0.067]
Activation ΔAcc	0.117 [0.082, 0.150]	0.064 [0.024, 0.104]
Median top1-top2 margin	4.65 nats	3.12 nats

The original total oracle and decoding effects are positive in Qwen3. Qwen's total oracle $D_1 - D_0$ effect is positive in all six folds and its interval excludes zero, while Goedel's locked Stage-C value is roughly 2.4 times larger with non-overlapping intervals (Table 3). Activation ΔAcc is also larger in Goedel. This is weaker than transferring Q1's identity-only reconstruction result: Stage C did not rerun $D_1 - D_C$ for Qwen. Nor is the model comparison fully symmetric, because the fixed candidate sets were partly proposed by Goedel. Qwen's preferences over those sets are also less decisive (median top1-to-top2 margin 3.12 versus 4.65 nats). The supported conclusion is therefore limited to positive total-oracle and activation-decoding effects in the parent under this shared-candidate evaluation; we do not identify specialization as the cause of the gap.

5 Discussion: From Explanations to Executable Interfaces

A different contract for activation interpretation. The central contribution of Veritas is to separate semantic definition from neural correspondence. The external executor defines a small set of candidate denotations before the activation reader is evaluated; the experiments then ask whether neural state predicts that identity, whether supplying the identity improves reconstruction, whether candidate-conditioned reconstructions steer behavior, and whether the constrained channel carries unrelated nuisance information. On the selected Lean states, the core measurements consistently show that externally fixed candidate identity is predictive, supports target-specific steering, and constrains the tested nuisance channel relative to freer alternatives. This is stronger than a free-form explanation that merely reconstructs well, while remaining short of the much stronger claim that the model's native computation is exactly the discrete variable Z.

Scope of the current evidence. The main limitations are concrete rather than conceptual. The dataset is drawn from successful Lean-accepted proofs and retains states only when the sampled trajectory is covered, even though the primary preferred-candidate label does not require that filter; only 109 of 244 MiniF2F test theorems therefore contribute records. The primary target uses length-sensitive total candidate log-probability at a late layer, so harder same-family candidates and alternative scoring rules would sharpen the semantic interpretation. The reconstruction decomposition depends on hashed text features, and the Qwen comparison reuses candidate sets partly proposed by Goedel. These limits define the next validation steps; they do not change the core result that externally fixed candidate identity has measurable predictive, intervention, and leakage consequences in the present system.

Toward verifier-grounded white-box control. Formal methods and activation interpretability are converging on a shared problem: how to connect neural computation to semantics that can be trusted outside the model. Formal methods make specified behavior checkable; activation methods expose model-internal predictive structure. VERITAS composes the two into an interface: an executable system defines the available meanings, and the neural model supplies evidence about which meaning is active. That interface is directly actionable. A sandbox can define file-system and test effects for a coding agent; a transaction simulator can define database effects; a typed API can define tool effects; a simulator can define embodied outcomes. In each case the monitor's vocabulary is not free-form text but a small set of externally checkable counterfactual effects.

This creates a concrete research program for AI safety and control. A verifier-grounded monitor can be evaluated not only for interpretability, but for detection latency, false-positive cost, adversarial evasion, and closed-loop intervention. White-box control and ControlArena provide natural settings for those experiments (AI Security Institute, 2025; Inglis et al., 2025; Kutasov et al., 2025). The next question is no longer whether a model can produce a plausible explanation of its activation; it is whether an externally grounded semantic interface remains informative and controllable when the agent is actively trying to defeat it.

6 Conclusion

Veritas introduces executable semantics as a design primitive for activation interpretability. Instead of asking a learned reader to invent the meaning of its own bottleneck, we first construct a finite set of externally executed candidate effects and then test how neural state relates to that fixed vocabulary. In Lean, candidate identity adds held-out activation-prediction value beyond the candidate-set decoder, the activation-conditioned pointer outperforms strong non-activation baselines, candidate-conditioned reconstructions produce target-specific steering, and the grounded pointer carries far less of the tested nuisance signal than an equal-cardinality free code. The same total-oracle reconstruction and decoding signals are positive in the Qwen3 parent under the shared candidate environment. More broadly, VERITAS points to a future in which formal verification and activation interpretability meet at an executable semantic interface: meanings are defined outside the model, while prediction, monitoring, and intervention are learned from within it.

Reproducibility Statement

The appendix documents the validation layer sweep, coverage and selection mechanism, seven pipeline bugs, nuisance calibration, language-bottleneck controls, matched-parent validation, and known limitations. The main text specifies theorem-level splitting and the primary candidate-scoring, reconstruction, decoding, intervention, and leakage evaluations. Exact executable replication additionally depends on the project code, model and Lean environment pins, random seeds, raw transition data, and per-stage result ledgers; these are separate artifacts rather than part of the manuscript itself.

AI Use Statement

Generative AI tools were used extensively as research assistants in this project. They helped refine hypotheses and experimental controls, review methodology, implement and debug parts of the research code, analyze experimental outputs, search and summarize related literature, and draft and edit portions of this manuscript. The authors executed the reported experiments on real models and data, manually reviewed resulting logs and claims, and independently validated critical AI-assisted code changes through tests, fresh Lean replay, or reruns before relying on them. Several methodological flaws reported in the paper were surfaced during AI-assisted review and then verified experimentally. The authors take responsibility for the final content, claims, code, and artifacts.

References

AI Security Institute. Why we are working on white box control. AI Security Institute blog, 2025. URL https://www.aisi.gov.uk/blog/why-were-working-on-white-box-control.

Jan Bauer, Celeste De Schamphelaere, Adam Karvonen, Niclas Luick, and Neel Nanda. Building better activation oracles. arXiv preprint arXiv:2606.02609, 2026.

Tobias Bersia and Tatiana Gaintseva. When activation oracles learn not to read: Concept-specific blind spots in fine-tuned oracles. arXiv preprint arXiv:2607.23379, 2026.

Caleb Biddulph and Adam Kaufman. Untrusted advice for AI control: Short, strong advice significantly uplifts weak LLMs. Redwood Research, 2026.

Varun Dhanraj and Chris Eliasmith. Improving rule-based reasoning in LLMs using neurosymbolic representations. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 30589–30608. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1556.

Haocheng Duan, Yuxin Guo, Jieyi Bi, Anqi Xie, Sirui Li, Yining Ma, and Cathy Wu. Interpreting neural combinatorial optimization via evolving programmatic bottlenecks. arXiv preprint arXiv:2606.19741, 2026.

Kit Fraser-Taliente, Subhash Kantamneni, Euan Ong, Dan Mossing, Christina Lu, Paul C. Bogdan, Emmanuel Ameisen, James Chen, Dzmitry Kishylau, Adam Pearce, Julius Tarng, Alex Wu, Jeff Wu, Yang Zhang, Daniel M. Ziegler, Evan Hubinger, Joshua Batson, Jack Lindsey, Samuel Zimmerman, and Samuel Marks. Natural language autoencoders produce unsupervised explanations of LLM activations. Transformer Circuits, 2026. URL https://www.transformer-circuits.pub/2026/nla/.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 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, pp. 160–187. PMLR, 2024.

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

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pp. 16295–16336. PMLR, 2024.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. Transformer Circuits, 2026. URL https://www.transformer-circuits.pub/2026/workspace/.

Itamar Hadad, Guy Katz, and Shahaf Bassan. Formal mechanistic interpretability: Automated circuit discovery with provable guarantees. In International Conference on Learning Representations, 2026.

Vincent Huang, Dami Choi, Daniel D. Johnson, Sarah Schwettmann, and Jacob Steinhardt. Predictive concept decoders: Training scalable end-to-end interpretability assistants. arXiv preprint arXiv:2512.15712, 2025.

Rogan Inglis, Ollie Matthews, Tyler Tracy, Oliver Makins, Tom Catling, Asa Cooper Stickland, Rasmus Faber-Espensen, Daniel O'Connell, Myles Heller, Miguel Brandao, Adam Hanson, Arathi Mani, Tomek Korbak, Jan Michelfeit, Dishank Bansal, Tomas Bark, Chris Canal, Charlie Griffin, Mojmir Stehlik, Jasmine Wang, and Alan Cooney. Controlarena. GitHub repository, 2025. URL https://github.com/UKGovernmentBEIS/control-arena.

ZeBin Ji, Yang Hu, Xiuli Bi, Bo Liu, and Bin Xiao. Select, hypothesize and verify: Towards verified neuron concept interpretation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2026.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, and Samuel Marks. Activation oracles: Training and evaluating LLMs as general-purpose activation explainers. arXiv preprint arXiv:2512.15674, 2025.

Shashank Kirtania. Activation steering in neural theorem provers. arXiv preprint arXiv:2502.15507, 2025.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pp. 5338–5348. PMLR, 2020.

Jon Kutasov, Chloe Loughridge, Yuqi Sun, Henry Sleight, Buck Shlegeris, Tyler Tracy, and Joe Benton. Evaluating control protocols for untrusted AI agents. arXiv preprint arXiv:2511.02997, 2025.

Tobias Ladner and Matthias Althoff. The misery of mechanistic interpretability: A formal perspective. arXiv preprint arXiv:2609.15533, 2026.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In International Conference on Learning Representations, 2026.

Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. arXiv preprint arXiv:2412.07992, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Kaiyu Yang, Aidan Swope, Alex Gu, et al. LeanDojo: Theorem proving with retrieval-augmented language models. In Advances in Neural Information Processing Systems, 2023.

Michael Zhang and Alex Turner. How robust are natural language autoencoders to initialization? TurnTrout research post, 2026. URL https://turntrout.com/natural-language-autoencoder-robustness.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: A cross-system benchmark for formal olympiad-level mathematics. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A Validation-only layer sweep

On 12 development theorems, the theorem-level ridge reconstruction gate was largest at the latest tested layer: layer 18, bootstrap mean 0.031 CI $[0.011, 0.064]$; layer 24, 0.025 $[0.004, 0.062]$; layer 26, 0.049 $[0.024, 0.085]$; layer 30, 0.051 $[0.028, 0.087]$. The sweep is not monotonic because layer 24 is below layer 18. We selected layer 30 before the full analysis. Because these 12 theorems were also used during pipeline development, this is a configuration-selection result, not an independent replication.

B Coverage and selection

Across 2,642 tactic-boundary candidate-construction attempts, 1,666 retained transitions had the sampled trajectory successor represented in the final candidate set, for 63.1% coverage. Atomic tactics are usually easy to cover; compound have transitions are not. HAVE accounts for 1,054 attempts but only 264 covered transitions (25.0%), because matching its successor requires independently proposing a specific intermediate proposition and proving it. The current preference experiment inherits this trajectory-coverage filter even though the preferred label $\arg\max_{c \in C} p(c \mid \text{prefix})$ is defined whenever C is nonempty. Removing that filter would eliminate this particular selection mechanism; whether the resulting states are harder or otherwise different is an empirical question.

C Audit: seven pipeline bugs found and fixed

The following were found through schema checks, source/activation alignment audits, or fresh Lean replay, and were fixed before the main locked analysis except where explicitly disclosed.

1. The pinned Lean REPL returned no tactic list in file mode; candidate tracing was moved to command mode.
2. REPL stdout was fully buffered against a plain pipe; a pseudo-terminal was required for reliable streaming.
3. Generated proofs omit the prompt preamble; reattaching it required corresponding source-offset correction.
4. Pretty-printed compound tactics did not literally match source substrings; alignment now trusts Lean's reported source position as a fallback.
5. The initial candidate generator inserted the observed tactic before truncation, creating severe label leakage. Candidate generation and truncation are now trajectory-independent.
6. <;> can cause Lean to report the same generated source span for multiple goals. Records are deduplicated by theorem and generated-text offset.
7. The REPL client could block indefinitely on either a full stdin pipe or blocking readline(). Reads and writes now run behind explicit deadlines with process recycling after timeout.

D Additional robustness details

The original uncalibrated nuisance injection used three bits at $\alpha = 0.15$ and produced near-zero leakage for both formal and arbitrary encoders. This was not interpretable evidence about steganography. We calibrated the operating point using the arbitrary-code encoder only, freezing two bits at $\alpha = 0.8$ once that positive control could reliably transmit nuisance information. Across five seeds, marginal leakage averaged 0.044 bits for the VERITAS pointer and 0.920 bits for the arbitrary code. A separate conditional evaluation trained attackers with access to (S, C, Z) and compared them to blind (S, C) attackers; VERITAS incremental accuracy stayed near zero, while arbitrary-code incremental accuracy was about 0.25. Repeating with linear, small-MLP, and large-MLP attackers changed the VERITAS gap only from 0.002 to 0.008 and left the arbitrary gap between 0.25 and 0.28.

E Matched-parent validation details

Before trusting Qwen3 numbers, the matched-parent extraction path was run on Goedel and reproduced stored activations at least 0.9999 cosine similarity and the locked Stage-C total oracle ΔFVE (0.1244) and activation ΔAcc (0.11668). For Qwen3, 28 of 30 permutation-null fold-seed combinations clear $p < 0.01$ and all 30 clear $p < 0.05$; zero and fixed-noise controls remain at or below context-only performance. Candidate sets are inherited from the Goedel pipeline, but Qwen preferred labels and margins are recomputed from Qwen's own candidate scores. Qwen's weaker top1-to-top2 margin (3.12 versus 4.65 nats) and the Goedel-conditioned candidate environment are unresolved confounds for interpreting the between-model gap.

F Additional limitations

Candidate generation is not exhaustive, canonicalization is conservative rather than theorem-proving equivalence, and the study covers one benchmark and one formal system. The primary target uses length-sensitive total candidate log-probability, so stronger evaluation on deliberately hard, same-family candidate sets would better isolate semantic information from late-layer lexical prediction. Finally, the state and candidate features are hashed text features; replication with alternative featurizations would test representation dependence.

G Vision: verifier-grounded white-box control

The VERITAS interface is not inherently Lean-specific. It requires three components: a state S, a finite set of counterfactual actions, and an external executor that maps each action to an observable successor. In code, the executor could be a sandbox plus unit tests and filesystem diffs; for tools, typed APIs plus effect logs; for agentic workflows, a sandboxed environment recording resource access, privilege changes, network egress, and other policy-relevant effects. A future white-box control protocol could map pre-action activations to externally checked action effects, use the code as an online monitor, and intervene by deferring, resampling, or steering toward an allowed candidate. Unlike Lean, agent environments may be stochastic, partially observed, expensive to fork, or unsafe to execute counterfactually; handling those cases is an open systems problem. ControlArena and related AI-control benchmarks provide natural adversarial evaluation environments for this extension (Kutasov et al., 2025; Inglis et al., 2025). The central open question is whether grounding a monitor's latent vocabulary in externally executed effects makes it more robust to adaptive evasion than an unconstrained activation probe at comparable rate and predictive power.