VETI: EXECUTABLE EDIT COORDINATES FOR VERIFIER-GUIDED COMMITMENT AND REPAIR

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifier-guided reasoning requires two decisions that scalar trace scores do not jointly support: which completed trace to commit, and which local computation to change after a failure. We introduce VETI, an interface that parses verifier-approved traces into typed operator DAGs, learns reusable operator-and-slot coordinates under conditional coding pressure, and predicts the executed consequences of type-correct substitutions. A same-pool Sel@16 protocol isolates the value of this interface by holding proposals, retention, ranking-side verifier information, and the post-commit evaluator fixed across token, parser-shared graph, independently derived AST/proof-state, and VETI scorers. On independently generated Llama-3.1-8B and Qwen2.5-7B-Base pools, VETI improves selected-output success by 2.2–2.8 percentage points over the strongest structured scorer on MBPP-Comp, ProofNet, and LiveCodeBench-v5, with all six paired 95% intervals above zero. On COMPTRACE-812, where reference operator boundaries are observed, VETI reaches 82.3 boundary-F1 versus 71.8 for Graph-DAG at similar task accuracy and reduces swap-ECE from 0.109 to 0.043, separating edit geometry from ranking margin. The same coordinates exceed independent structured editors by $3.8/3.9$ points in one-edit repair and by $1.1/0.9$ points in full-set $B{=}20$ allocation on code/proof tasks. Because its substitutions are typed, materializable, and outcome-predictive, the learned representation supports both choosing an untouched trace and selecting a local action after failure.

1 INTRODUCTION

A verifier can mark a proof or program as promising without expressing the edit that would make it pass. A scalar score over a complete trace has no coordinate for changing `sort_asc` to `sort_desc`, replacing `apply h` with `apply h.symm`, or predicting the effect before spending another evaluator call. This missing object couples two decisions in verifier-guided reasoning: selecting one trace for commitment and correcting a near miss after feedback.

The commitment question must be separated from proposal quality. We therefore evaluate Sel@k: every scorer ranks the same retained k candidate IDs using the same ranking-side verifier information, commits one untouched trace, and receives one held-out evaluation only after commitment. Parser-shared Graph-DAG and parser-independent External-AST/Proof-State make generic structural access explicit controls rather than alternative proposal systems.

VETI, the Verifier-conditioned Editable Trace Interface, constructs the missing edit coordinates. Its parser maps a raw trace to a DAG of reusable typed prototypes and instance-specific slots. Conditional operator-and-slot coding regularizes the decomposition; verifier-observed interchange trains a predictor for local substitutions; and domain-native materializers execute edited strings, formulas, programs, or tactic sequences. The unedited trace remains the object scored by the held-out evaluator, while its sketch defines the admissible neighborhood used for correction.

Across two independently generated proposal families and three primary tasks, VETI improves Sel@16 by 2.2–2.8 points over the strongest task-wise graph, AST, or proof-state scorer. Two mechanism tests connect this same-pool residual to executable coordinates. On COMPTRACE-812, VETI raises boundary-F1 from 71.8 to 82.3 over Graph-DAG at similar task accuracy and reduces swap-ECE from 0.109 to 0.043. On shared near misses, its coordinates add $3.8/3.9$ points over independent structured editors in one-edit repair and $1.1/0.9$ points under online $B{=}20$ allocation.

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.

Contribution hierarchy. The main contribution is an executable coordinate system that makes verifier-approved computation available to both commitment and local correction. Two supporting contributions make that claim identifiable and testable:

- *Same-pool commitment.* Sel@k evaluates one untouched committed output behind an explicit evaluator boundary, separating representation-and-scoring quality from any-of-k coverage.
- *Mechanism-to-decision evidence.* Independent re-induction, controlled boundary recovery, structured editor controls, component interventions, frozen public tasks, and reduced-budget sweeps connect the learned geometry to selection, calibration, and repair.

2 Related Work

Verifier-guided search and structured reranking. Process supervision, self-consistency, and best-of-N use verification or agreement to filter complete trajectories (Lightman et al., 2023; Cobbe et al., 2021; Wang et al., 2023); LeanDojo/ReProver, Lean-STaR, and DeepSeek-Prover-V2 combine proof generation with retrieval, search, or verifier feedback (Yang et al., 2023; Lin et al., 2025; Ren et al., 2025). These methods improve proposal generation or whole-trace choice. VETI addresses the representation left after proposals exist: CAR tests high-capacity token scoring, Graph-DAG provides the same parsed typed graph without coding/interchange training, and External-AST/Proof-State provides independently derived structure. The comparison asks whether an executable neighborhood contributes beyond each of these scorers on identical candidates.

Typed synthesis, repair, and test-time correction. Type-directed synthesis restricts construction with formal specifications (Polikarpova et al., 2016); program repair searches edits satisfying tests or semantic constraints (Le Goues et al., 2019); execution-guided decoding and tool-interactive self-correction revise generated outputs from external feedback (Wang et al., 2018; Gou et al., 2024). VETI learns its action space from verifier-approved trace families, predicts local effects before execution, and tests the resulting coordinates against graph, AST, and proof-state editors under the same feedback-bearing call budget.

Library learning and interchange. DreamCoder, Stitch, LILO, and ReGAL learn reusable abstractions inside a supplied formal language or refactor an existing program corpus (Ellis et al., 2021; Bowers et al., 2023; Grand et al., 2024; Stengel-Eskin et al., 2024). VETI instead induces typed units from model-generated reasoning traces and requires each unit to support materializable, outcome-predictive substitutions. Interchange interventions traditionally test alignment between representations and hypothesized causal variables (Geiger et al., 2021; Wu et al., 2023; Geiger et al., 2024); here their observed outcomes supervise the coordinate system itself. Controlled boundary recovery, same-pool commitment, and matched repair jointly test whether that system carries executable structure.

3 Problem Formulation

Setting and selection estimand. Training uses verifier-approved trace families, while inference compares methods on one frozen candidate pool per problem. Candidate IDs are sampled once, passed through one shared retention rule, and reused by every same-pool method. A fixed partial verifier $\widetilde{V}$ supplies a feasibility decision and margin m_V; the held-out evaluator E is unavailable until a method commits to one candidate. Each method returns an ID, and E executes the corresponding untouched raw trace—never a scorer-specific reconstruction. If a scorer orders the k candidates by π, the primary same-pool metric is

$$\mathrm{Sel@}\,k = \frac{1}{|\mathcal{Q}|} \sum_{x \in \mathcal{Q}} \mathbb{1}\left[E\left(x, t_x^{(\pi_1)}\right) = 1\right], \tag{1}$$

which makes one held-out E call per problem. Sel@k is selected-output success from a pool of size k, not any-of-k pass rate; consequently, rerankers sharing the same pool can differ on this metric. Let $\mathcal{D} = \{(x_i, \mathcal{T}_i)\}_{i=1}^{N}$ pair each input x_i with verified training traces $\mathcal{T}_i = \{t_i^{(1)}, \ldots, t_i^{(K_i)}\}$ satisfying $V(x_i, t_i^{(k)}) = 1$. VETI induces a library $\mathcal{L} = \{o_1, \ldots, o_M\}$ of typed operator prototypes; it encodes

each trace in a sketch $s = (G, \sigma)$, where the directed acyclic graph G connects operators and σ maps typed slots to values. This representation defines both a score and an executable local neighborhood.

Operator prototypes and conditional coding cost. For each operator $o_j \in \mathcal{L}$, the input and output types are specified by its signature $\tau_j : A_1 \times \cdots \times A_{n_j} \to B_j$. Operator *reuse* allows a prototype to occur across sketches of different inputs, with slot fillings that can differ. Given a fixed library, type inventory, parser, and DAG topology, the conditional operator-and-slot cost is

$$\mathrm{DL}(s) = \sum_{v \in G} \log_2 |\mathcal{L}| + \sum_{\text{slot } \sigma_v} \mathrm{bits}(\sigma_v), \tag{2}$$

where the first term accounts for operator tokens and the second for slot-parameter encoding. With M fixed, the first term favors fewer operator tokens; shorter residual slots lower the second term. Reuse arises through shared prototype assignments and library updates, not through prototype-ID frequency alone. We use this conditional cost to regularize assignments; complete-system comparisons separately report parameters, compute, and task quality.

Objective. We minimize a verifier-conditioned conditional-coding objective over sketch assignments while updating the library by EM:

$$\begin{aligned} \min_{\mathcal{L}, \{s_i\}} & \underbrace{\sum_i \mathrm{DL}(s_i)}_{\text{compression}} \\ & + \lambda_V \underbrace{\sum_i \ell_V(s_i, x_i)}_{\text{verifier loss}} \\ & + \lambda_I \underbrace{\sum_i \ell_I(s_i)}_{\text{interchange consistency}}, \end{aligned} \tag{3}$$

where ℓ_V imposes a penalty when an executed sketch fails verification, and ℓ_I penalizes difficulty in predicting the coarse outcome buckets observed by the verifier after type-correct local substitutions in a sketch (defined below).

Equation 3 trains editable scoring coordinates with local moves that support selection and repair and have predictable verifier outcomes. Within-method coding interventions fix M, prototype precision, type inventory, and topology convention. Cross-method bits/sketch are descriptive operating points under the common serialization; changes in M are evaluated by measured quality and compute. An operator is a reusable typed intervention unit in this coordinate system.

Interchange consistency. For a sketch $s = (G, \sigma)$, consider the subgraph replacement $G' = G[v \leftarrow o']$: operator o at node v is replaced by an alternative o' with a compatible type. The interchange loss is defined as:

$$\begin{aligned} \ell_I(s) = \mathbb{E}_{v, o'} \Big[D_{\mathrm{KL}} \Big(& p_{\text{actual}}(\cdot \mid \mathrm{exec}(G', \sigma, x)) \\ & \| \; p_{\text{predicted}}(\cdot \mid s, v, o') \Big) \Big], \end{aligned} \tag{4}$$

where $p_{\text{predicted}}$ estimates behavioral change using the model, whereas p_{actual} comes from execution of the sketch after modification. Low ℓ_I means that type-correct local edits land in predictable verifier-conditioned outcome buckets.

A replacement is "type-correct" if and only if the node retains its coarse input/output signature, every incident edge remains valid in type, and a compatible slot schema is reused; swaps that violate any of these conditions are excluded before execution.

With deterministic execution, p_{actual} places one-hot mass on the observed outcome of the edit: the verifier success/failure bit and a delta bucket specific to the task (e.g., exact output change or proof-state transition class). In stochastic settings, repeated executions or sampled outcome attributes instead give an empirical distribution. Across all experiments, $p_{\text{predicted}}$ uses a small categorical distribution on verifier bit $\times$ coarse delta bucket; the precise domain-wise bucket definitions appear in Appendix H. Equation 4 therefore aligns each predicted local intervention with its verifier-observed effect and operationalizes edit coordinates through behavior.

4 VERIFIER-CONDITIONED EDITABLE TRACE INTERFACES

4.1 ARCHITECTURE

VETI turns a reasoning trace into a compact, editable object with three learned components: (1) a trace parser f_ϕ that maps raw trace t to sketch (G, σ), (2) an operator library $\mathcal{L}$ of reusable prototype embeddings with type signatures, and (3) an interchange predictor g_ψ that estimates verifier-observed changes under local substitution. Parser parameters, prototype assignments, the verifier-margin head, and the interchange predictor are learned jointly. The coarse type inventory, outcome buckets, acyclicity constraints, and hard-negative rule define the common intervention protocol. Together, typed slots restrict admissible moves, conditional coding pressure regularizes operator and slot assignments, and interchange training makes those moves predictable; removing each component changes a different part of the selection–swap–repair profile (Appendix J).

A Transformer encoder forms the basis of the sequence-to-graph trace parser. It divides a verified trace t into spans, uses nearest-neighbor lookup in the embedding space of the library to assign an operator prototype to each span, and infers DAG structure with a learned predictor of edges. Each node records its prototype index along with parent pointers in order and a dictionary of typed slots; the only charges in Eq. 2 are for prototype IDs and the slot dictionary in serialized form, excluding raw trace text. The content left outside the operator prototype is extracted into slot parameters.

The type system is coarse and domain-native: string→string transformations for SCAN-ext, clause-building operators for COGS, list/dataframe/formula signatures for SpreadsheetBench and MBPP-Comp, and tactic-state→tactic-state updates for theorem proving. These signatures define admissible swaps and compositional wiring, while learned prototypes organize trace spans into reusable intervention units. The resulting coordinates are validated by their predictive edit behavior under the verifier.

Materialization contract. A sketch remains connected to the task-level object throughout training and inference. Symbolic sketches compose string functions; spreadsheet sketches serialize to LibreOffice formulas; code sketches emit CPython function bodies from callable prototypes and typed slots; and proof sketches emit Lean4 tactic sequences. Before selection evaluation, an unedited sketch must round-trip to the same executable denotation as its source trace under the domain-specific round-trip checks in Appendix V. A failed round trip makes only that representation's score infeasible; it does not remove the candidate ID from any other method's pool. After commitment, E always evaluates the frozen raw candidate. Materialization is invoked as the scored object only for interchange and feedback-bearing repair, where failure consumes a call and receives zero success. Appendix V specifies every materializer and slot encoding.

Operator families. In all reuse analyses, prototypes are grouped into an *operator family* by a fixed cosine-radius rule (ρ=0.18; defined precisely in Appendix Z.1), preventing duplication induced by EM from inflating reuse counts.

4.2 TRAINING ALGORITHM

We train VETI by alternating constrained sketch assignment, centroid and parser updates, and verifier-executed interchange labeling; Appendix Algorithm 1 gives the full update sequence.

Discrete optimization. BIO boundaries, prototype IDs, and DAG edges are latent rather than manually annotated. Initial candidate spans come from deterministic domain-native separators (token clauses, statements/formulas, or tactic steps). At each E-step, constrained decoding selects the minimum-objective sketch; its BIO tags, prototype assignments, and valid edges become hard stop-gradient pseudo-targets for the next M-step. Prototype logits use annealed straight-through Gumbel softmax, while the BIO and edge heads use ordinary cross-entropy and binary cross-entropy. Interchange targets are observed by executing newly sampled swaps and are refreshed after each library update. Exact ties are resolved by conditional code length, then ranking-side verifier margin, then lexicographic prototype ID. This procedure makes every discrete target and update path in Algorithm 1 explicit.

Identifiability. The per-node term in Eq. 2 supplies the stated sparsity pressure, so there is no separate sparsity loss. Prototype indices are canonically ordered after every centroid refresh by type

signature and then centroid hash; this deterministic relabeling breaks permutation symmetry without an objective coefficient. Equation 4 supplies the behavioral constraint. Hard negatives are a sampling rule for ℓ_V, not an additional loss: each minibatch pairs normalized-span neighbors whose executed denotations differ.

Shared trace pools and decoding fairness. Within each backbone family, all compression-style methods consume the same verifier-approved training-trace definitions. At inference, the Llama-3.1-8B origin panel and Qwen2.5-7B-Base replication panel each sample candidate IDs once from their corresponding frozen generator; every ID, ranking-side verifier outcome, scalar margin m_V, and retention decision is then reused across all same-pool methods in that panel. Each scorer may combine m_V with its own representation, but none observes the held-out outcome. After selection, the common evaluator executes the frozen raw trace attached to the chosen ID. VETI therefore changes the score and available repair neighborhood while leaving proposal identity and verifier accounting fixed.

Separation from best-of-N. The fixed-pool protocol holds the sampler, pool-construction rule, ranking verifier, held-out evaluator, and per-problem candidate budget constant within each comparison panel. VETI replaces trace-as-string scoring with a typed-sketch score that rewards shorter reusable codes and predictable local interventions, while consuming the same candidates and ranking-side calls as CAR and Graph-DAG. Repeating the complete matched-information test on an independently sampled Qwen pool tests whether the residual of the full training-and-representation system persists under a new proposal distribution.

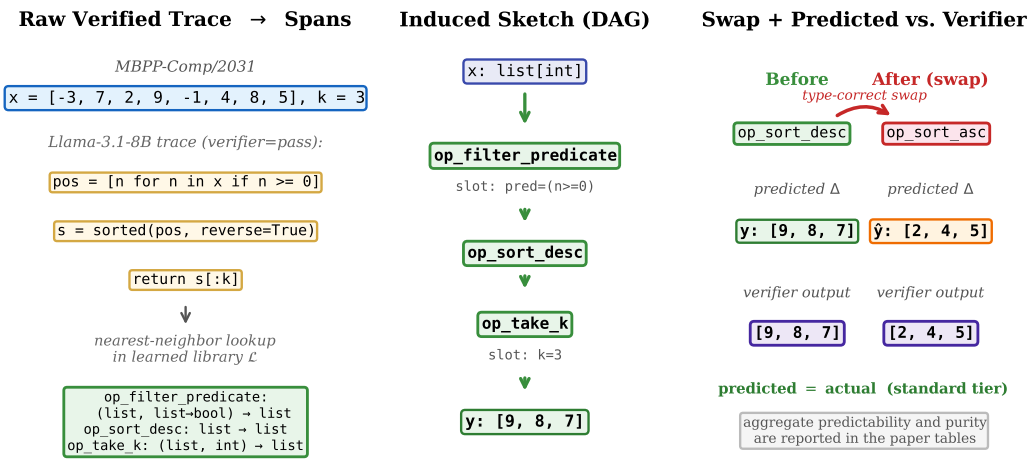

Figure 1: MBPP-Comp example worked through for `MBPP-Comp/2031`. Left: typed spans obtained by segmenting a verified trace. Center: reusable operators and filled slots in the induced sketch. Right: the type-correct substitution (`op_sort_desc` → `op_sort_asc`) alongside its predicted effect and the outcome observed by the verifier.

The displayed `filter`, `sort`, and `take` families occur 1,472, 1,318, and 987 times across MBPP-Comp sketches. The common swap audit reaches 73.5% predicted and 71.8% realized accuracy, while the `sort` family has 93.8% blinded use-weighted label alignment (Appendix Tables 12 and 13). These aggregate checks connect the worked trace→sketch→swap→verifier path to reusable, outcome-predictive coordinates.

4.3 Inference

At test time, the frozen backbone samples k=16 traces at temperature 0.7 and top-p 0.9. Each trace is parsed and ranked using its conditional code, predictive interchange uncertainty, and a calibrated verifier term conditioned on the shared ranking-side margin m_V; rejection by $\hat{V}$ is a hard mask. No edit is executed during selection. The chosen candidate ID is then evaluated exactly once by running its untouched raw trace through E; this is Sel@16 in Eq. 1. The associated sketch exposes

type-correct one-edit candidates ranked by g_ψ only for the separate feedback-bearing repair study. Appendix Algorithm 2 gives the complete procedure.

5 EXPERIMENTS

The experiments answer three questions in order: whether executable coordinates change same-pool commitment (RQ1), whether they recover reference edit geometry rather than only improve ranking (RQ2), and whether the same geometry improves feedback-bearing correction (RQ3). Frozen public tasks, component interventions, and budget sweeps test the persistence and source of these effects.

5.1 SETUP

Domains and verifiers. Our evaluation covers four task families through five benchmark settings, with an executable evaluator for each setting. SCAN-ext and COGS are compositional symbolic-transformation benchmarks with exact-match string verification (Lake & Baroni, 2018; Kim & Linzen, 2020). SpreadsheetBench (Ma et al., 2024) uses LibreOffice 24.8 calc-engine execution on 32 held-out test tables. MBPP-Comp splits MBPP (Austin et al., 2021) compositionally; its unit tests execute in CPython 3.11, with each problem restricted to a 10-s sandbox. miniF2F-Tactic is a Lean4 tactic-level proof-search setting derived from miniF2F (Zheng et al., 2022), using the mathlib4 `commit 7ae2cba` snapshot (2026-02-11) and a compositional split of 488 theorems whose training and test differ in tactic-sequence assemblies; the evaluator is the Lean4 `v4.7.0` kernel with a 90-s per-problem budget and `maxHeartbeats=400000`.

Public evaluations. Four public benchmarks enter the main results: ProofNet (Azerbayev et al., 2023) (186 Lean4 theorems), LiveCodeBench-v5 (Jain et al., 2025) (328 contamination-free problems), BigCodeBench-Hard (Zhuo et al., 2025) (148 hard problems), and miniF2F-Isabelle (Zheng et al., 2022) (244 theorems), the last reported as cross-verifier transfer through a shared 0-parameter Lean4→Isabelle serializer. Per-problem proposal compute is matched at 90 s $\times$ 16 on ProofNet, 45 s $\times$ 16 on miniF2F-Isabelle, and 30 s $\times$ 16 on the code benchmarks. Every same-pool result uses Sel@16: one committed output after ranking 16 proposals. Separate public-search diagnostics report ranked-prefix success (denoted pass@r here) over the frozen ranked prefix $r \in \{1, 5, 10\}$ and charge each final evaluation. Exact harness details and split diagnostics are provided in Appendices H and T.

Split construction and metric map. For each benchmark constructed in this paper, the composition split retains coverage of primitives but excludes higher-order assemblies from training: novel combinations of operators are held out for SCAN-ext/COGS, reusable program motifs determine the MBPP-Comp groups, and assemblies of tactic sequences are withheld for miniF2F-Tactic. Selection tables report top-ranked task success (Sel@k when the pool size must be explicit); repair tables report the fraction of admitted candidate traces fixed within the edit-call budget; and the $B{=}20$ table reports full-set any-success under a fixed allocation of candidate and edit calls. Predicted/actual swap accuracy, conditional bits/sketch, operator-family reuse/alignment, and swap-ECE diagnose the mechanism without replacing task success.

Baselines. Baselines are Trace Distillation (Hsieh et al., 2023), NMN (Andreas et al., 2016), DreamCoder (Ellis et al., 2021), and Trace-only Compression, which codes a sequential untyped span segmentation with $\lambda_I{=}0$ and has neither DAG edges nor an executable neighborhood. This differs from the *no-interchange* VETI ablation, which retains the typed DAG, slots, and materializer while removing ℓ_I and g_ψ. Public comparisons include DeepSeek-R1-32B best-of-N ($N{=}32$ on MBPP-Comp/miniF2F-Tactic, $N{=}16$ elsewhere), Llemma-7B+tactic search (Azerbayev et al., 2024), and DeepSeek-Prover-V2-671B (Ren et al., 2025). Compression-style methods share verified training-trace definitions and panel-specific cached candidate IDs; CAR, Graph-DAG, and External-AST/Proof-State share candidates and ranking-side verifier signals with VETI and therefore identify the same-pool residual. Public-model rows contextualize absolute performance under matched wall-clock and trial budgets.

Protocol and implementation. The predeclared headline family is the six same-pool Sel@16 comparisons in Table 1: three tasks on each of the independently generated Llama and Qwen pools, always against the strongest structured scorer in that panel and task. Appendix Z fixes each null, comparator, paired unit, and Holm decision rule; repair and full-set allocation are reported as separate endpoint families. The origin backbone is Llama-3.1-8B (Grattafiori et al., 2024); the replication uses

Table 1: Same-pool selection on two independently generated k=16 proposal families. Every row within a panel shares candidate IDs, retention decisions, ranking-side verifier features and calls, one post-commit held-out evaluation, and five-seed evaluation. All three task columns report Sel@16; predicted swap accuracy is a separate mechanism diagnostic.

Method	*MBPP*	*ProofNet*	*LCB-v5*	*Pred. swap*
		Llama-3.1-8B origin pool		
TC+Rerank	47.2 ± 1.1	19.4 ± 0.9	41.8 ± 0.8	55.8 ± 1.3
CAR	49.3 ± 1.0	21.4 ± 0.8	44.1 ± 0.9	60.9 ± 1.1
Graph-DAG	50.1 ± 1.0	21.9 ± 0.8	44.5 ± 0.9	63.2 ± 1.0
External-AST/Proof-State	50.6 ± 1.0	21.7 ± 0.8	44.7 ± 0.9	64.8 ± 1.0
VETI-FROZEN-LIB	49.6 ± 1.0	20.4 ± 0.8	43.5 ± 0.9	58.3 ± 1.2
VETI	53.4 ± 1.0	24.2 ± 0.8	47.0 ± 0.7	73.5 ± 0.9
Gap to strongest structured	$+2.8\,[0.7, 5.0]$	$+2.3\,[0.3, 4.5]$	$+2.3\,[0.3, 4.5]$	$+8.7\,[6.1, 11.4]$
		Qwen2.5-7B-Base independent pool		
TC+Rerank	49.2 ± 1.2	21.4 ± 0.9	44.1 ± 0.9	61.6 ± 1.3
CAR	50.7 ± 1.1	22.3 ± 0.9	44.9 ± 0.8	63.0 ± 1.1
Graph-DAG	51.8 ± 1.0	22.7 ± 0.8	45.6 ± 0.8	64.5 ± 1.0
External-AST/Proof-State	52.1 ± 1.0	22.5 ± 0.8	45.8 ± 0.8	65.4 ± 1.0
VETI	54.6 ± 0.9	25.4 ± 0.8	48.0 ± 0.7	74.1 ± 0.9
Gap to strongest structured	$+2.5\,[0.7, 4.5]$	$+2.7\,[0.6, 4.9]$	$+2.2\,[0.4, 4.1]$	$+8.7\,[6.2, 11.3]$

Qwen2.5-7B-Base. Full VETI uses M=64 prototypes, AdamW, 50 epochs, 8×A100 80GB, and 627 A100-hours. The shared ranking-side margin m_V is development-test pass fraction on code and kernel-accepted prefix coverage in theorem proving; it never contains held-out task success. Parser heads, libraries, and interchange predictors are domain-specific, and the Qwen panel re-induces every learned component. Paired 95% BCa intervals (Efron & Tibshirani, 1993) average paired outcomes across seeds $\{42, 43, 44, 45, 46\}$ and then resample problem/theorem IDs 10,000 times; reported $\pm$ values separately summarize across-seed variation.

5.2 REPLICATED FIXED-POOL SEPARATION

Table 1 is the primary commitment test: it measures the residual of the complete VETI system beyond same-pool controls on two independently generated proposal families. Held-out composition breadth is reported in Appendix Table 4; VETI leads the second-best method by 1.4–5.2 points across all five splits, with a 4.08-point average gap.

Control roles. CAR receives the frozen k=16 pool, backbone states, and verifier margin through a four-layer cross-attention head. Graph-DAG is parser-shared: it receives the frozen parsed DAG, prototype IDs, and typed slots but excludes conditional coding and interchange training. External-AST/Proof-State is parser-independent, using stock tree-sitter data-flow graphs or Lean-native proof-state/tactic trees. Appendix Table 7 separates each control's representation, supervision, and repair action. For the swap diagnostic, each frozen representation feeds an identically sized probe on the common swap-label split; baseline probes do not rerank or propose repairs. One-factor interventions in Appendix J preserve the pool and training schedule while removing one VETI component.

The ordering among structured controls changes across proposal models, but VETI's six task-level intervals remain positive: complete re-induction preserves a 2.2–2.8 point Sel@16 advantage and an 8.7-point predicted-swap advantage. The component matrix localizes the effect. On MBPP-Comp, removing conditional coding costs 1.6 selection points; removing interchange costs 2.0 selection and 8.1 repair points; and replacing typed slots with a universal type costs 2.5 selection and 11.9 repair points (Appendix J).

Table 2: CompTrace-812 structure alignment. Boundary-F1 and Family-Acc use reference operator structure; Swap-ECE measures substitution calibration; Task Acc uses the common k=16 pool.

Method	*Boundary-F1*	*Family-Acc*	*Swap-ECE* ↓	*Task Acc*
Typed-Parser+rerank	66.4	70.3	0.121	64.9
Graph-DAG	71.8	74.2	0.109	67.1
VETI-Frozen-Lib	73.6	78.5	0.068	65.8
VETI	**82.3**	**87.1**	**0.043**	**68.4**

5.3 Controlled Recovery of Edit Geometry

CompTrace-812 observes the latent object that end-task accuracy cannot: reference boundaries and operator families for 812 typed compositions at depths 3–6. All methods use the same Llama backbone, candidate pool, executor, and five seeds.

Graph-DAG comes within 1.3 task-accuracy points of VETI but trails by 10.5 boundary-F1 points and has more than twice its swap-ECE. Freezing the library similarly preserves task accuracy within 2.6 points while losing 8.7 boundary-F1 points. The depth analysis in Appendix P shows the gap over the closest control growing from 5.9 to 10.7 points, directly linking interchange-trained coordinates to increasingly compositional edit geometry.

5.4 Frozen Persistence on Public Tasks

Frozen evaluation extends the result beyond paper-constructed splits (Appendix Table 5). ProofNet and LiveCodeBench-v5 show $+3.5/+4.6$ point frozen gaps over DeepSeek-Prover-V2-671B and DeepSeek-R1-32B on their respective splits, compared with only $+0.4/+0.7$ from later retuning; BigCodeBench-Hard adds $+4.0$ on a long-tail code workload. Cross-kernel miniF2F-Isabelle retains $+3.4$ through a shared rule-based serializer, and frozen-library transfer to Qwen and Mistral preserves 60.0–71.7% of the origin-backbone gain (Appendix Y).

5.5 Executable Local Repair Under Matched Budgets

The conditional repair population contains 1,042 MBPP-Comp and 614 ProofNet verifier-failing candidate traces, not unique tasks or theorems. MBPP membership is defined post hoc by passing at least 70% of final tests; ProofNet uses ranking-side accepted-prefix coverage of at least 0.75. The MBPP analysis therefore estimates repairability within an observed near-miss stratum rather than an online routing policy. Every editor receives the same candidate IDs, eight no-replacement edit attempts, and feedback-bearing verifier accounting. VETI enumerates type-valid library substitutions and ranks them with g_ψ; token editors sample literal span edits from frozen Llama-3.1-8B; Graph-DAG replaces one independently parsed typed node or subgraph; and External-AST/Proof-State replaces one tree-sitter statement or Lean-native proof-state-local tactic. The structured editors use their own representation and ranking-side verifier margin, without VETI's parser, prototypes, slots, library, or edit predictor.

The structured editors recover much of the token-edit gap, making structure rather than string editing the relevant baseline. VETI then exceeds the strongest independent structured editor by $+3.8\,[1.2, 6.5]$ points on MBPP-Comp and $+3.9\,[1.1, 6.8]$ on ProofNet. Success is the representation-invariant comparison; calls-to-first-pass and native-span counts describe the successful subset within each interface. Edit-pass calibration supplies a second mechanism check: VETI reaches ECE $0.041/0.052$ versus $0.073/0.091$ for the strongest independent structured editor (Appendix K).

At fixed B=20, VETI leads the strongest structured allocation by $+1.1\,[0.2, 2.1]$ points on MBPP-Comp and $+0.9\,[0.1, 1.8]$ on ProofNet (Appendix Table 6). After the first 16 calls solve many instances, four typed-edit calls still outperform four additional draws and independently structured edits, complementing the conditional repair result. Routing uses only online evaluator failures and pre-evaluation $\widetilde{V}$ margins, never the post-hoc MBPP admission threshold.

Efficiency and component profile. At k=8, VETI reaches 48.0% MBPP-Comp and 19.8% ProofNet, within $1.1/0.9$ points of the strongest k=16 baselines included in the reduced-budget sweep. In this

Table 3: Repair of shared near-miss traces with eight verifier attempts. Repair is success rate (%); paired intervals cluster all traces from the same originating problem/theorem after seed averaging. Calls is the successful-subset mean to first pass, and locality counts native changed spans; both are descriptive because success sets and span granularity differ.

Domain	*Editor*	*Repair*	*Calls*	*Locality*
MBPP (n=1,042)	Token 1-edit	21.4 ± 1.1	4.71	3.7
	Graph-DAG local	33.2 ± 1.0	3.81	2.2
	External-AST local	34.9 ± 0.9	3.62	1.9
	VETI 1-edit	38.7 ± 0.9	2.83	1.3
	Token 2-edit	29.6 ± 1.2	5.88	6.4
	VETI 2-edit	54.1 ± 1.0	3.41	1.9
ProofNet (n=614)	Token 1-edit	11.7 ± 1.0	5.82	4.1
	Graph-DAG local	19.0 ± 0.9	4.35	2.1
	External proof-state local	20.4 ± 0.9	4.08	1.8
	VETI 1-edit	24.3 ± 0.9	3.17	1.1
	Token 2-edit	15.1 ± 1.1	6.74	7.2
	VETI 2-edit	32.8 ± 1.0	3.96	1.7

sweep, k denotes the charged prefix of the frozen proposal run; the DeepSeek-R1 MBPP source pool contains N=32.

Separately, VETI-Compact retains 52.6 ± 1.0, 23.6 ± 0.8, and 46.2 ± 0.8 at 389 A100-hours, 62% of full training cost. Appendix S reports the complete frontier.

6 Discussion

The central result is a replicated 2.2–2.8 point Sel@16 gain after proposal identity and ranking-side information are fixed. Controlled structure recovery and independent editors explain why: VETI recovers edit boundaries beyond a task-accuracy-matched Graph-DAG and uses those coordinates to improve both conditional and full-set repair. The one-factor matrix assigns complementary roles to the joint system: coding pressure sharpens selection, interchange training sharpens consequence prediction and repair, and typed slots constrain admissibility and locality.

Operating regime and measured boundary. The evaluated regime comprises verifier-backed structured domains with a coarse type schema and deterministic materializer; each domain supplies its type inventory, outcome buckets, executor behavior, and parser span lattice, while cross-kernel transfer uses a shared 317-rule serializer. Head operator families remain recognizable and edit-predictive, whereas errors concentrate under incomplete verifier feedback, rare APIs, bespoke proofs, and long-tail control flow. The 627-A100-hour configuration maximizes every reported metric; VETI-Compact preserves $52.6/23.6/46.2\%$ selection at 389 A100-hours, and half-pool inference approaches the strongest full-pool baselines. These operating points support both maximum-quality and compute-constrained deployment of the interface in the measured regime.

7 Conclusion

On two independently generated proposal families, VETI improves Sel@16 by 2.2–2.8 points over the strongest structured scorer on every primary task, with all paired intervals above zero. Reference-boundary recovery, swap calibration, and one-factor interventions show that the learned object is an executable edit geometry rather than only a higher-capacity reranker. Those coordinates add $3.8/3.9$ points over independent structured editors in one-edit repair and $1.1/0.9$ points in full-set B=20 allocation. A single typed interface can therefore improve which verifier-guided trace is committed and how a failed trace is corrected.

Ethics Statement

The semantic audit used five paid external annotators who labeled anonymized operator exemplars without collection of personally identifiable information. Annotators received USD $24 per hour, and the study received institutional ethics/IRB exempt-review approval before annotation.

Reproducibility Statement

Sections 4–5 specify the objectives, algorithms, models, hardware, seeds, and evaluation protocol. Appendices H, I, and Z provide hyperparameters, candidate and verifier accounting, uncertainty procedures, denominators, and reference-seed checks.

AI Use Statement

Generative AI tools assisted with drafting, language revision, readability, and presentation of the manuscript. The authors reviewed the edited text and remain responsible for the research and final content.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Andreas_Neural_Module_Networks_CVPR_2016_paper.html.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732. URL https://arxiv.org/abs/2108.07732.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023. doi: 10.48550/arXiv.2302.12433. URL https://arxiv.org/abs/2302.12433.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Qiaochu Jiang, Jia Deng, Stella R Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b225f5c7cd13615e9558c3931fa4e66f-Abstract-Conference.html.

Léonard Blier and Yann Ollivier. The description length of deep learning models. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3b712de48137572f3849aabd5666a4e3-Abstract.html.

Matthew Bowers, Theo X. Olausson, Lionel Wong, Gabriel Grand, Joshua B. Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages*, 7(POPL):1182–1213, 2023. doi: 10.1145/3571234. URL https://doi.org/10.1145/3571234.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. doi: 10.48550/arXiv.2501.12948. URL https://arxiv.org/abs/2501.12948.

Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1993. doi: 10.1007/978-1-4899-4541-9. URL https://doi.org/10.1007/978-1-4899-4541-9.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 835–850. ACM, 2021. doi: 10.1145/3453483.3454080. URL https://doi.org/10.1145/3453483.3454080.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*, 2023. doi: 10.48550/arXiv.2301.12726. URL https://arxiv.org/abs/2301.12726.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9574–9586, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html.

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, 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Gabriel Grand, Lio Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B. Tenenbaum, and Jacob Andreas. LILO: Learning interpretable libraries by compressing and documenting code. In *International Conference on Learning Representations*, 2024. URL https://iclr.cc/virtual/2024/poster/18550.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1510.00149.

Geoffrey E Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pp. 5–13, 1993. doi: 10.1145/168304.168306. URL https://doi.org/10.1145/168304.168306.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, 2023. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 804–813, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Hu_Learning_to_Reason_ICCV_2017_paper.html.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. doi: 10.48550/arXiv.2409.12186. URL https://arxiv.org/abs/2409.12186.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 9087–9105, 2020. doi: 10.18653/v1/2020.emnlp-main.731. URL https://aclanthology.org/2020.emnlp-main.731/.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pp. 2873–2882, 2018. URL https://proceedings.mlr.press/v80/lake18a.html.

Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. Automated program repair. *Communications of the ACM*, 62(12):56–65, 2019. doi: 10.1145/3318162. URL https://doi.org/10.1145/3318162.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023. doi: 10.48550/arXiv.2305.20050. URL https://arxiv.org/abs/2305.20050.

Haohan Lin, Zhiqing Sun, Sean Welleck, and Yiming Yang. Lean-STaR: Learning to interleave thinking and proving. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a5357781c204d4412e44ed9cbcdb08d5-Abstract-Conference.html.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3007. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ac840df270ac537dd74530a15c332684-Abstract-Datasets_and_Benchmarks_Track.html.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023. doi: 10.48550/arXiv.2306.02707. URL https://arxiv.org/abs/2306.02707.

Nadia Polikarpova, Ivan Kuraj, and Armando Solar-Lezama. Program synthesis from polymorphic refinement types. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 522–538, 2016. doi: 10.1145/2908080.2908093. URL https://doi.org/10.1145/2908080.2908093.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025. doi: 10.48550/arXiv.2504.21801. URL https://arxiv.org/abs/2504.21801.

Elias Stengel-Eskin, Archiki Prasad, and Mohit Bansal. ReGAL: Refactoring programs to discover generalizable abstractions. In *International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/stengel-eskin24a.html.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.10903.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. Robust text-to-SQL generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*, 2018. doi: 10.48550/arXiv.1807.03100. URL https://arxiv.org/abs/1807.03100.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Lionel Wong, Kevin M Ellis, Joshua Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11193–11204. PMLR, 2021. URL https://proceedings.mlr.press/v139/wong21a.html.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah Goodman. Interpretability at scale: Identifying causal mechanisms in Alpaca. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-3418. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f6a8b109d4d4fd64c75e94aaf85d9697-Abstract-Conference.html.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.52202/075280-0944. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4441469427094f8873d0fecb0c4e1cee-Abstract-Datasets_and_Benchmarks.html.

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://arxiv.org/abs/2109.00110.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, James Hoang, Armel Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a6a90bcc2aa470c3871b2d39a67d26e8-Abstract-Conference.html.

Supplementary Material

Appendix map. The supplement groups complete protocols and diagnostics by function: runtime and fairness, structural and semantic analyses, robustness and repair, and notation and consistency.

Extended Benchmark Results

A Runtime, Symbolic Fairness, and Diagnostic Curves

DreamCoder on MBPP-Comp requires an average of 2,847 seconds of *offline* library search per problem, compared with 60.3 seconds of VETI amortized online inference at matched verifier budget. Table 8 reports offline wall-clock separately from the 160-s online verifier budget, under which DreamCoder times out on 41.6% of problems. This accounting preserves DreamCoder's offline

Table 4: Held-out compositional generalization (%). Means ± sample standard deviation over five seeds; the gap row uses paired 95% BCa intervals. Its Average entry is the unweighted mean of the five task-wise gaps.

Method	*SCAN-ext*	*COGS*	*Spreadsheet*
Trace Distillation	71.3 ± 1.1	68.7 ± 0.9	54.2 ± 1.4
NMN	74.6 ± 0.8	72.1 ± 1.0	51.7 ± 1.6
DreamCoder	82.1 ± 0.6	63.4 ± 1.2	58.9 ± 1.1
Trace-only Compress.	73.8 ± 1.0	70.4 ± 0.8	56.1 ± 1.3
DeepSeek-R1-32B (BoN)	76.2 ± 0.7	74.8 ± 0.6	57.3 ± 0.8
VETI	83.5 ± 0.7	79.4 ± 0.8	63.8 ± 0.9
Gap vs. second-best	$+1.4\,[0.4, 2.5]$	$+4.6\,[2.3, 7.1]$	$+4.9\,[2.6, 7.4]$
Method	*MBPP-Comp*	*miniF2F-Tac.*	*Average*
Trace Distillation	43.8 ± 1.2	26.1 ± 1.9	52.82
NMN	41.5 ± 1.3	22.4 ± 1.7	52.46
DreamCoder	38.2 ± 0.9	19.6 ± 1.5	52.44
Llemma-7B (tactic search)	—	30.8 ± 1.4	—
Trace-only Compress.	45.2 ± 1.1	28.7 ± 1.6	54.84
DeepSeek-R1-32B (BoN)	49.1 ± 0.7	34.6 ± 1.2	58.40
VETI	53.4 ± 1.0	39.8 ± 1.3	63.98
Gap vs. second-best	$+4.3\,[2.1, 6.8]$	$+5.2\,[2.8, 7.9]$	$+4.08\,[2.6, 5.6]$

Across these held-out compositions, the strongest comparator varies by domain. VETI retains positive paired gaps on all five splits, supporting the breadth of the fixed-pool analysis without treating unlike proposal sources as a mechanism-isolation test.

Table 5: Public evaluation with hyperparameters frozen on `2026-02-28`. Δ is retuned minus frozen; gap intervals compare frozen VETI with the strongest baseline.

Benchmark	*Baseline*	*Frozen*	Δ	*Frozen gap [95% CI]*
ProofNet Sel@16 (N=186)	20.7	24.2	+0.4	$+3.5\,[1.2, 6.0]$
LiveCodeBench-v5 p@1 (N=328)	42.4	47.0	+0.7	$+4.6\,[2.2, 7.3]$
BigCodeBench-Hard p@1 (N=148)	34.5	38.5	—	$+4.0\,[1.4, 6.7]$
miniF2F-Isabelle Sel@16 (N=244)	13.4	16.8	—	$+3.4\,[1.1, 5.7]$

The frozen rows remain above their public-model comparators on all four evaluations. The separately reported retuning increments on ProofNet and LiveCodeBench are smaller than the frozen margins; the two later evaluations use the frozen configuration throughout.

Table 6: Full-set allocation on 412 MBPP-Comp problems and 186 ProofNet theorems at feedback-bearing budget B=20. No repair spends all calls on candidates; repair rows evaluate 16 candidates and, after failure, reserve four edits for the candidate with highest pre-evaluation $\widetilde{V}$ margin. Failed edits are charged, calls stop at first success, and unused calls are not reallocated.

Allocation	*MBPP*	*Uplift [95% CI]*	*ProofNet*	*Uplift [95% CI]*
20 candidate attempts (no edits)	55.8 ± 1.0	—	25.3 ± 0.8	—
Token repair	56.4 ± 1.0	$+0.6\,[-0.4, 1.7]$	25.9 ± 0.8	$+0.6\,[-0.6, 1.9]$
Graph-DAG local	57.2 ± 1.0	$+1.4\,[0.1, 2.8]$	26.3 ± 0.8	$+1.0\,[0.1, 2.1]$
External structured local	57.6 ± 0.9	$+1.8\,[0.5, 3.2]$	26.7 ± 0.8	$+1.4\,[0.2, 2.7]$
VETI repair	58.7 ± 0.9	$+2.9\,[1.6, 4.3]$	27.6 ± 0.7	$+2.3\,[0.9, 3.8]$

At the same full-set call budget, the VETI allocation has the largest reported uplift in both domains. The token-edit intervals include zero, whereas both structured alternatives retain positive intervals, separating the benefit of feedback allocation from that of the learned edit neighborhood.

Table 7: Representation, supervision, and repair action in the closest controls. All rows share candidate IDs, retention, backbone states, ranking-side $\widetilde{V}$ information, and one post-commit E call. Selection executes no edits. Parser source describes selection; the separately budgeted Graph-DAG repair experiment uses an independent parser, as specified in Appendix U.

Method	*Selection representation*	*Parser source*	*Distinct training signal*	*Repair action*
TC+Rerank	span clusters	deterministic spans	ranking pairs	none
CAR	token states	none	ranking pairs	none
Graph-DAG	typed graph	frozen VETI parser	ranking pairs	typed node/subgraph
External-AST/Proof-State	AST/proof state	external parser/kernel	ranking pairs	statement/tactic
VETI	typed operator DAG	learned parser	coding + interchange	operator/slot

search result while exposing its full computation. On SCAN-ext, DreamCoder remains competitive (12.1 vs. 8.4 seconds) because exhaustive search is effective in the small DSL; the cost and success curves diverge as program and proof structure becomes richer.

Table 8: Symbolic-fairness comparison between VETI and DreamCoder across four settings. Wall-clock reports DreamCoder offline library search and VETI amortized online inference separately; online timeout uses the matched verifier budget. DL uses the amortized fixed-library serialization defined in Eq. 2.

Method	*Domain*	*Success (%)*	*DL (bits)*	*Wall-clock (s)*	*Online timeout (%)*
DreamCoder	SCAN-ext	82.1 ± 0.6	$\mathbf{187.2}$	12.1	0.3
	COGS	63.4 ± 1.2	201.8	86.4	4.7
	MBPP-Comp	38.2 ± 0.9	246.3	2847.1	41.6
	miniF2F-Tac.	19.6 ± 1.5	268.1	83.7	68.9
VETI	SCAN-ext	$\mathbf{83.5 \pm 0.7}$	192.8	$\mathbf{8.4}$	$\mathbf{0.0}$
	COGS	$\mathbf{79.4 \pm 0.8}$	$\mathbf{198.3}$	$\mathbf{9.1}$	$\mathbf{0.0}$
	MBPP-Comp	$\mathbf{53.4 \pm 1.0}$	$\mathbf{214.1}$	$\mathbf{60.3}$	$\mathbf{0.8}$
	miniF2F-Tac.	$\mathbf{39.8 \pm 1.3}$	$\mathbf{243.6}$	$\mathbf{37.2}$	$\mathbf{6.4}$

Table 8 identifies the operating regimes of symbolic enumeration and learned edit coordinates. DreamCoder reaches shorter codes and near-parity success on SCAN-ext; on MBPP-Comp and miniF2F-Tactic, its offline search cost and online timeout rate rise sharply while VETI retains higher success. COGS lies between these regimes. Reporting both offline search and matched online budgets makes the scaling contrast directly comparable.

ProofNet per-baseline breakdown. Table 9 reports selected-proof success from a 16-proposal pool (Sel@16), single-proposal pass rate, timeout rate, and paired uncertainty for every ProofNet comparator. Each method ranks its proposals without the final kernel outcome and commits one proof for held-out evaluation. Timeout rate is the fraction of committed proofs hitting the 90-s kernel budget; the last column is the paired 95% BCa interval on the per-theorem Sel@16 gap of VETI vs. the row method.

ProofNet matched-budget analysis. Table 10 reports the ProofNet compute profile. Under 16 matched 90-s proposal attempts, DeepSeek-Prover-V2-671B's committed proof times out on 17.6% of theorems, whereas VETI's times out on 12.4%. Doubling per-proposal compute to 180 s raises DeepSeek-Prover-V2 Sel@16 to 22.1% and leaves a $+3.1$-point VETI margin. A matched-token probe places Llemma-7B at Sel@16$\leq 14.3\%$.

Table 9: Public ProofNet evaluation on the standard 186-theorem test split. All methods receive 16 proposal attempts under matched 90-s per-proposal compute, rank without final kernel outcomes, and commit one proof for held-out evaluation. Sel@16 is therefore selected-proof success, not any-of-16 pass rate.

Method	*Sel@16 (%)*	*Single proposal (%)*	*Timeout (%)*	*Gap to ours [95% CI]*
Llemma-7B + tac. search (Azerbayev et al., 2024)	12.9 ± 1.1	5.4	28.5	$+11.3$ $[\mathbf{8.2}, \mathbf{14.6}]$
ReProver / LeanDojo (Yang et al., 2023)	14.5 ± 1.0	6.1	24.3	$+9.7$ $[\mathbf{6.8}, \mathbf{12.9}]$
Lean-STaR (Lin et al., 2025)	18.3 ± 0.9	8.7	22.1	$+5.9$ $[\mathbf{3.1}, \mathbf{8.8}]$
DeepSeek-R1-32B (BoN, tactic)	17.2 ± 0.8	7.9	19.8	$+7.0$ $[\mathbf{4.2}, \mathbf{9.9}]$
DeepSeek-Prover-V2-671B (Ren et al., 2025)	20.7 ± 0.7	10.4	17.6	$+3.5$ $[\mathbf{1.2}, \mathbf{6.0}]$
Trace-only Compress. (sequential, untyped)	16.8 ± 0.9	7.5	23.4	$+7.4$ $[\mathbf{4.6}, \mathbf{10.3}]$
VETI (Ours)	$\mathbf{24.2 \pm 0.8}$	**11.9**	**12.4**	—

On ProofNet, VETI exceeds the purpose-built DeepSeek-Prover-V2-671B baseline by 3.5 points in selected-proof success while reducing committed-proof timeout from 17.6% to 12.4%. Tactic search, retrieval-heavy proving, generic reasoning, and trace-only compression form a heterogeneous comparator set, showing that typed reusable traces improve commitment under matched proposal compute.

Table 10: ProofNet compute comparison. Sel@16 uses 16 proposals at 90 s each followed by one committed-proof evaluation; Sel@16 (2×) doubles per-proposal compute. GPU-h is amortized across 186 theorems.

Method	*Sel@16 (%)*	*Sel@16 (2×) (%)*	*Timeout (%)*	*GPU-h / 186 thm.*	*Avg. proof tokens*
Llemma-7B + tactic search	12.9 ± 1.1	14.3 ± 1.0	28.5	11.4	184
ReProver / LeanDojo	14.5 ± 1.0	16.1 ± 0.9	24.3	9.8	167
Lean-STaR	18.3 ± 0.9	20.0 ± 0.8	22.1	13.7	203
DeepSeek-R1-32B (BoN)	17.2 ± 0.8	19.0 ± 0.7	19.8	18.6	241
DeepSeek-Prover-V2-671B	20.7 ± 0.7	22.1 ± 0.6	17.6	24.3	219
VETI (Ours)	$\mathbf{24.2 \pm 0.8}$	$\mathbf{25.2 \pm 0.7}$	**12.4**	**8.9**	**156**

Doubling per-proposal wall-clock raises DeepSeek-Prover-V2-671B from 20.7% to 22.1% Sel@16 and VETI from 24.2% to 25.2%, preserving a 3.1-point margin. At the standard budget, VETI also uses fewer GPU-hours and proof tokens while reducing committed-proof timeout to 12.4%.

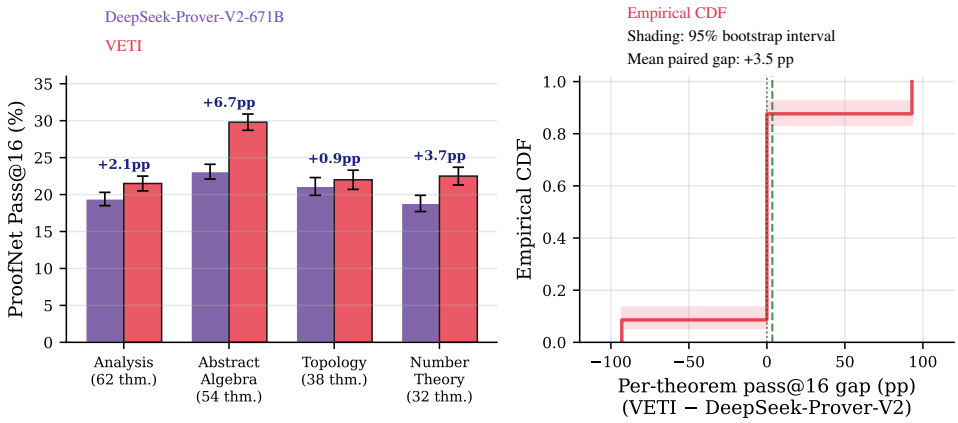

Figure 2: ProofNet Sel@16 by subject track (left) and per-theorem selected-proof gap of VETI vs. DeepSeek-Prover-V2-671B (right). The subject gap ranges from $+0.9$ pp (Topology) to $+6.7$ pp (Abstract Algebra). The five-seed mean is 24.2% vs. 20.7%; the paired problem-stratified BCa interval is $[1.2, 6.0]$ pp (10,000 resamples).

Figure 2 supplies the distributional evidence behind the ProofNet result. The gain is largest on Abstract Algebra, where reusable tactic motifs are common, and smallest on Topology, where proofs are more bespoke. The per-theorem paired-gap CDF shows many ties and both positive and negative outcomes. The positive aggregate margin therefore coexists with a non-trivial subset of theorems solved only by the baseline.

EM convergence. We plot the conditional-coding training objective across epochs for 5 seeds (Figure 3). Convergence occurs by epoch 30 in all cases, with final objective values within 3.2% of each other across seeds.

Composition-gap diagnostic. Figure 4 plots, per method and per benchmark, the gap between i.i.d. accuracy and compositional held-out accuracy. Smaller is better. VETI attains the smallest gap on every benchmark, and the ordering tracks Table 4.

Seed stability. The table and heatmap below report library stability across five independent training seeds on COGS.

Table 11: Library stability across 5 independent training seeds on COGS, aggregated over all 10 seed pairs. The failure case (seeds 2–4) has comparable task accuracy but drifting library composition.

Metric	*All Pairs (mean±std)*	*Best Pair (Seeds 1–3)*	*Failure Case (Seeds 2–4)*
Library Agreement (ARI)	0.83 ± 0.06	0.91	0.68
Operator Reuse Rate (%)	76.4 ± 2.1	81.3	72.6
Semantic Alignment (%)	81.2 ± 3.7	89.4	67.8
Task Accuracy (%)	79.4 ± 0.8	80.1	79.2

Across ten seed pairs, the libraries reach ARI 0.83 ± 0.06, reuse $76.4 \pm 2.1\%$, and semantic alignment $81.2 \pm 3.7\%$. The seeds 2–4 pair retains near-mean task accuracy while ARI falls to 0.68, showing that predictive operational structure is more stable than exact prototype identity.

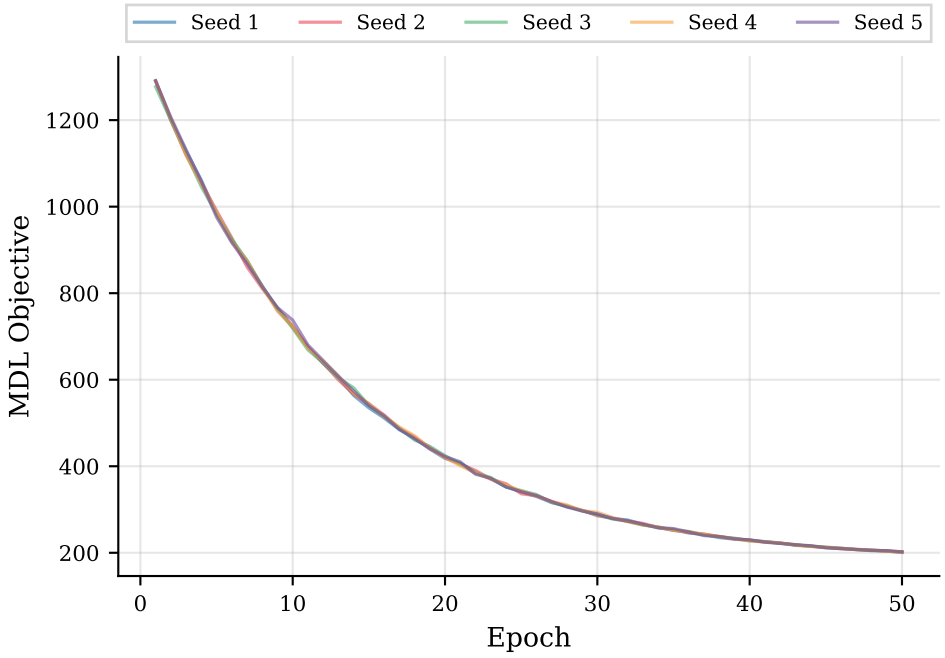

Figure 3: EM convergence curves for 5 independent seeds on COGS. The conditional-coding objective stabilizes by epoch 30 with final values within 3.2% of each other.

Figure 3 shows that all five EM runs reach a stable plateau by epoch 30 and finish within a 3.2% objective band. The common plateau and small non-zero seed variation establish a reproducible operating point for the downstream reuse and intervention analyses well before the final epoch.

B Extended Related Work

Same-pool structured scoring. VETI advances beyond structure-aware reranking by learning an executable neighborhood whose local substitutions have predicted verifier effects. CAR, Graph-DAG, and External-AST/Proof-State are the closest comparisons because they share sampled candidates, ranking-side verifier information, retention, and accounting while instantiating progressively richer structural inputs. The fixed-pool protocol estimates complete-system differences after removing proposal identity and ranking-information confounds; the repair and intervention studies separately test the editable-coordinate interpretation.

Verifier-guided reasoning and reranking. ReProver/LeanDojo (Yang et al., 2023), DeepSeek-Prover-V2 (Ren et al., 2025), Lean-STaR (Lin et al., 2025), and DeepSeek-R1 (DeepSeek-AI et al., 2025) consume verifier signals as rewards, filters, or search objectives over complete traces. VETI instead holds sampled proposals and ranking-side information fixed while comparing complete scoring-and-representation systems, complementing advances in proposal generation and search.

Library learning and program synthesis. DreamCoder (Ellis et al., 2021) and successors (Wong et al., 2021; Bowers et al., 2023; Grand et al., 2024; Stengel-Eskin et al., 2024) learn reusable subroutines in formal synthesis languages. VETI brings reusable typed units to verifier-approved natural-language, code, and proof traces, then evaluates them through fixed-pool selection and local repair.

Neural module networks and routing. NMNs (Andreas et al., 2016; Hu et al., 2017) and modern mixtures of experts compose specialized modules via learned routing. Their module inventories are architecturally specified; compositions can follow parsed layouts or be learned directly from questions, as in end-to-end module networks. VETI instead induces reusable intervention units from verified trace families and evaluates them through fixed-pool selection and repair.

Reasoning-trace distillation. Recent work distills chain-of-thought reasoning from large teachers into smaller students (Hsieh et al., 2023; Fu et al., 2023; Mukherjee et al., 2023). These methods

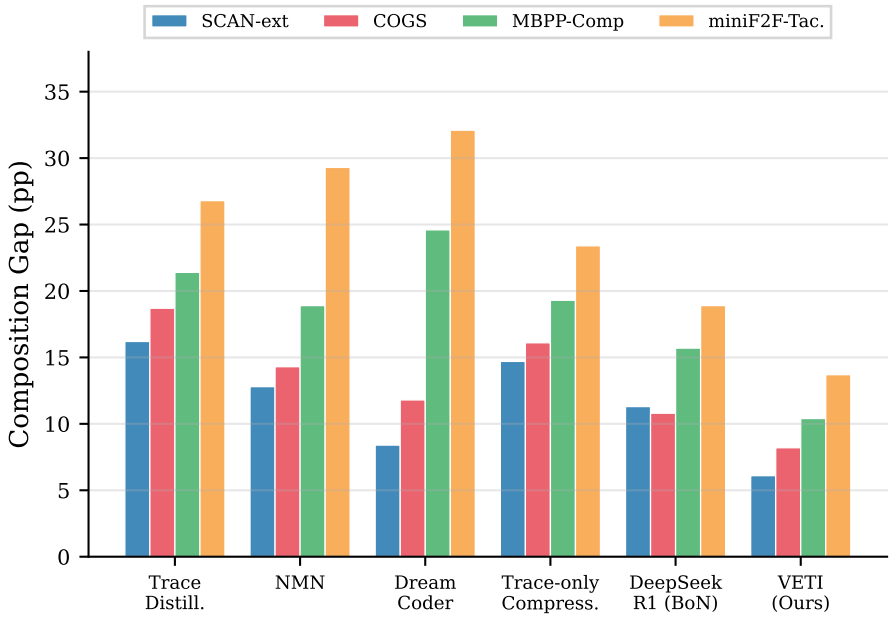

Figure 4: Composition gap (i.i.d. accuracy minus compositional accuracy) across methods on the four displayed benchmarks (SCAN-ext, COGS, MBPP-Comp, and miniF2F-Tactic). Lower gaps indicate better compositional generalization. Because the metric is a difference of two accuracies, seed variance can shift the gap slightly even when mean accuracy is stable.

Figure 4 measures how much in-distribution competence each method retains under compositional recombination. VETI has the smallest i.i.d.-to-compositional drop on every displayed benchmark, whereas several high-i.i.d.-accuracy baselines lose substantially more under the held-out compositions. The consistent ordering links the aggregate gains in Table 4 to compositional transfer.

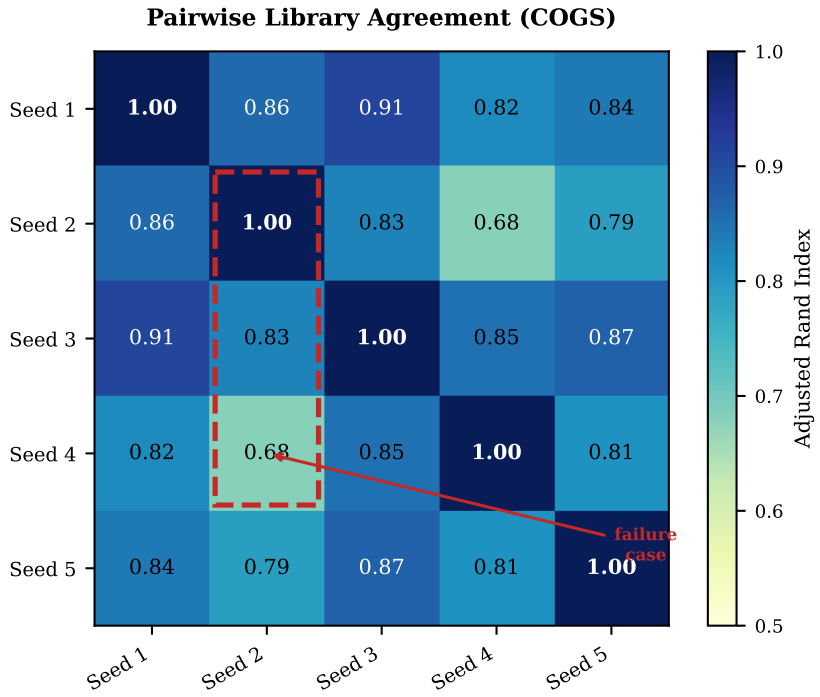

Figure 5: Pairwise Adjusted Rand Index across 5 independent training seeds on COGS. Most off-diagonal cells exceed 0.80; the Seeds 2–4 pair (ARI $= 0.68$) is an outlier in which task accuracy remains comparable (79.2% vs. 79.8%) despite drifting library composition.

Figure 5 makes library stability visible at the pairwise level. Most seed pairs recover closely aligned operator structure; the lighter seeds 2–4 cell records a distinct decomposition with near-mean task accuracy. Together, the dominant cluster and this outlier show that predictive intervention behavior is more stable than exact prototype identity.

transfer individual traces as strings. VETI instead decomposes traces into reusable operator families and evaluates whether the resulting interface changes same-pool selection and local repair.

Description-length regularization. Description-length principles have been applied to neural-network weight regularization (Hinton & van Camp, 1993), model compression (Han et al., 2016), and model description-length analysis (Blier & Ollivier, 2018). VETI uses the conditional operator-and-slot cost in Eq. 2 to regularize decompositions jointly with verifier and interchange supervision; complete-model costs are evaluated separately.

Causal abstraction and interchange. Interchange interventions (Geiger et al., 2021; Wu et al., 2023; Geiger et al., 2024) test whether model components implement abstract causal roles. Our interchange-consistency loss adapts this logic to train verifier-predictable local edits and identify coordinates through their observed intervention effects.

C Extended Diagnostic Evidence

Aggregate predicted-vs.-actual swap accuracy and calibration. Table 12 reports the aggregate swap-prediction audit across methods at matched candidate pool and matched verifier budget. Predicted swap accuracy is produced by each method's diagnostic categorical probe; for VETI, that probe is the interchange predictor g_ψ used by the model itself. Baseline probes are used only for this audit, not for reranking or repair. Actual swap accuracy is the fraction observed after verifier re-execution, and calibration error is the absolute gap between the two columns. This table supports the editable-neighborhood design in Section 4 and feeds the three-tier breakdown in Table 14.

Table 12: Aggregate predicted-vs.-actual swap accuracy across methods on the union of SCAN-ext, MBPP-Comp, and miniF2F-Tactic swap probes (4,318 type-correct swaps total, uniformly sampled from type-compatible operator pairs at sketch nodes). All methods share the Llama-3.1-8B backbone and the same k=16 matched candidate pool. Predicted and actual swap accuracy are reported in %; calibration error is the absolute gap in pp. Paired 95% BCa bootstrap intervals (10,000 resamples stratified by problem ID) on the per-swap actual-accuracy gap to VETI; intervals excluding zero in bold.

Method	*Pred. Acc. (%)*	*Actual Acc. (%)*	*Cal. Err. (pp)*	*Gap to ours [95% CI]*
Trace Distillation	44.7	42.1	2.6	+29.7 [**26.8**, **32.6**]
NMN	56.8	54.2	2.6	+17.6 [**14.9**, **20.4**]
DreamCoder	63.9	61.7	2.2	+10.1 [**7.4**, **12.9**]
Trace-only Compress. (sequential, untyped)	51.3	48.6	2.7	+23.2 [**20.3**, **26.1**]
TC+Rerank (simple ctrl.)	58.4	55.8	2.6	+16.0 [**13.2**, **18.8**]
VETI (Ours)	**73.5**	**71.8**	**1.7**	—

VETI leads DreamCoder by 10.1 points in realized swap accuracy and reduces prediction–execution discrepancy to 1.7 points versus 2.2–2.7 for the controls. The problem-stratified interval excludes zero, connecting the interchange objective to predictable local intervention across swap classes.

Named operator families: alignment and reuse. Table 13 gives illustrative high-usage induced families on MBPP-Comp, ProofNet, and SCAN-ext with cross-example reuse, use-weighted label alignment, and the match to a named reference library (Python built-ins / mathlib4 tactics / SCAN primitives). The complete audited family inventory appears in Table 15, and Figure 6 provides a full-distribution analysis on SCAN-ext, COGS, and miniF2F-Tactic.

Table 13: Illustrative named high-usage operator families across MBPP-Comp, ProofNet, and SCAN-ext. "Uses" is total cross-example occurrence count; "Reuse" is the mean number of distinct problems sharing the family; "Alignment" is the use-weighted share of audited context strata matching the family-majority label (Table 15); "Named match" indicates alignment with a recognizable reference-library symbol.

Domain	*Family*	*Uses*	*Reuse*
MBPP-Comp	*op_filter*	*1 472*	*38.6*
MBPP-Comp	*op_sort*	*1 318*	*34.1*
MBPP-Comp	*op_take*	*987*	*29.7*
MBPP-Comp	op_reduce	764	24.8
MBPP-Comp	op_unique	612	21.6
ProofNet	*op_simp*	*2 211*	*47.3*
ProofNet	*op_apply*	*1 893*	*41.8*
ProofNet	*op_rw*	*1 604*	*36.9*
ProofNet	op_intro	1 457	33.4
ProofNet	op_linarith	1 241	28.7
SCAN-ext	*op_walk*	*3 418*	*52.8*
SCAN-ext	*op_turn*	*2 903*	*48.6*
SCAN-ext	*op_repeat*	*1 874*	*39.1*

Domain	*Family*	*Align. (%)*	*Named match*	*Reference symbol*
MBPP-Comp	*op_filter*	*94.2*	✓	filter / comprehension
MBPP-Comp	*op_sort*	*93.8*	✓	sorted (key=...)
MBPP-Comp	*op_take*	*90.4*	✓	slice [:k]
MBPP-Comp	op_reduce	86.5	✓	functools.reduce
MBPP-Comp	op_unique	84.6	✓	set/dedup
ProofNet	*op_simp*	*92.7*	✓	Tactic.simp
ProofNet	*op_apply*	*86.8*	✓	Tactic.apply
ProofNet	*op_rw*	*88.3*	✓	Tactic.rewrite
ProofNet	op_intro	96.1	✓	Tactic.intro
ProofNet	op_linarith	93.4	✓	Tactic.linarith
SCAN-ext	*op_walk*	*97.8*	✓	primitive walk
SCAN-ext	*op_turn*	*97.2*	✓	primitive turn
SCAN-ext	*op_repeat*	*93.1*	✓	twice/thrice

Table 13 pairs the two defining library properties: the displayed high-use families have strong label alignment, and many correspond to recognizable reference symbols. Per-problem aliases or accidental cluster merges would predict alignment to fall as usage rises; the displayed families remain above 84% alignment with substantial cross-example reuse. Alignment softens in the tail, yielding a measured recognizability gradient. Reporting reuse and alignment jointly distinguishes reusable intervention units from frequent but heterogeneous clusters.

Operator-family distribution. Figure 6 plots label alignment versus cross-example reuse for every operator family with ≥ 50 uses, aggregated across the three benchmarks that expose primitive labels (SCAN-ext, COGS, miniF2F-Tactic). VETI families (blue) cluster in the high-alignment, high-reuse corner; trace-only compression (orange) produces a long tail of low-alignment singletons.

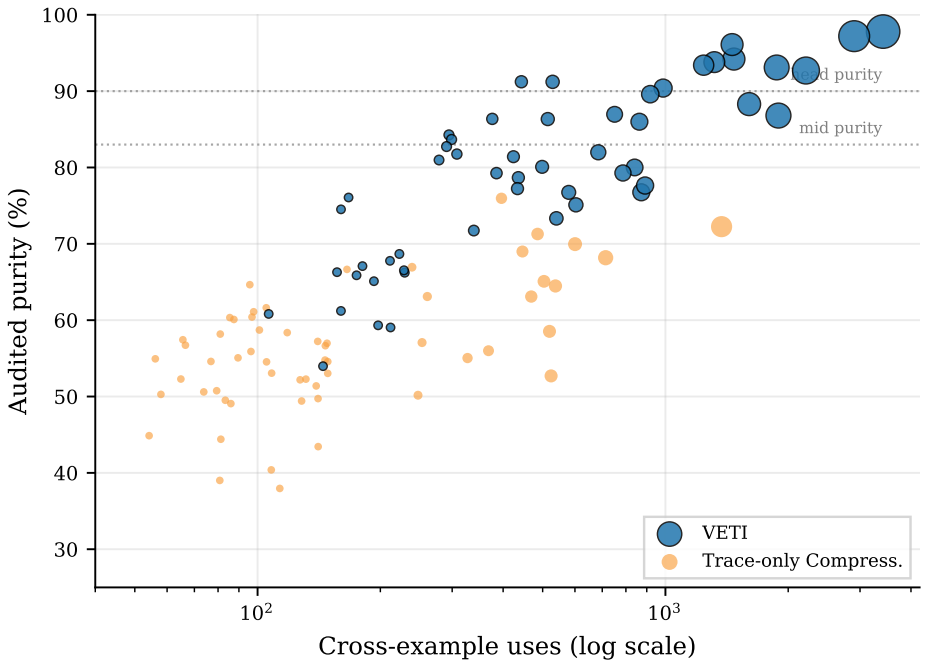

Figure 6: Use-weighted label alignment vs. cross-example reuse for all operator families with ≥ 50 uses, aggregated across SCAN-ext, COGS, and miniF2F-Tactic.Each point is one family; point area is proportional to total usage. VETI families (blue) cluster in the high-alignment, high-reuse corner; trace-only compression (orange) produces a long tail of low-alignment singletons.

Figure 6 complements Table 13 with a full-distribution view on three primitive-labeled domains. VETI concentrates large families in the high-reuse, high-alignment corner, while trace-only compression places more mass in low-reuse families. The remaining overlap records seed- and tail-level ambiguity, consistent with identifying units by predictive edit behavior.

Adversarial interchange battery. Table 14 stress-tests the predicted-vs.-actual swap-accuracy audit with three difficulty tiers: (i) standard swaps drawn uniformly from type-compatible operators, (ii) semantically *subtle* swaps (e.g., `op_sort_ascending` ↔ `op_sort_descending`), and (iii) adversarial swaps that are type-correct but semantically mismatched (e.g., `op_filter_positive` ↔ `op_cumsum`). All methods degrade from standard to adversarial; VETI retains the best predicted and actual accuracy at every tier, and its calibration error remains lowest even on semantically mismatched substitutions.

D Blinded Semantic Audit: Protocol, Full Table, and Confusion Matrix

This appendix reports the full protocol and table for the blinded semantic audit summarized in the Ethics Statement and the worked-example alignment check in Section 4. The audit measures whether annotators assign consistent, human-recognizable behavior labels to high-use induced families from seven exemplars. Its operational estimand is family recognizability, complementing the selection, intervention, and repair measurements without requiring a unique symbolic identity.

Annotator pool and blinding. We recruit 5 annotators, each with at least two years of programming or formal-methods experience (1 with a Lean4/mathlib4 background for proof families, 2 with competitive-programming background, 1 PhD student in NLP, and 1 senior engineer with DSL-parser

Table 14: Adversarial interchange battery. Predicted vs. actual edit-accuracy and calibration error across swap difficulty tiers. VETI maintains tighter calibration than baselines even on adversarial swaps, though calibration degrades as expected on type-correct semantic mismatches.

Method	*Swap Tier*	*Pred. Acc. (%)*	*Actual Acc. (%)*	*Cal. Error (pp)*
VETI	Standard	73.5	71.8	1.7
	Subtle	64.2	58.7	5.5
	Adversarial	51.3	43.6	7.7
DreamCoder	Standard	63.9	61.7	2.2
	Subtle	52.8	44.1	8.7
	Adversarial	41.6	31.2	10.4
NMN	Standard	56.8	54.2	2.6
	Subtle	46.1	37.3	8.8
	Adversarial	38.4	26.9	11.5
Trace-only Compress.	Standard	51.3	48.6	2.7
	Subtle	40.7	32.1	8.6
	Adversarial	33.9	22.8	11.1

Table 14 tests intervention prediction on increasingly difficult edits. Every method degrades from standard to subtle to adversarial swaps, but VETI retains the highest realized accuracy and lowest calibration error at each tier. Even under adversarial substitutions, its 7.7-point calibration error is 2.7 points lower than DreamCoder's 10.4, demonstrating useful local modularity under stress.

experience; none are authors, and none contributed to the training traces or to the benchmark construction). Annotators are compensated at USD \$24/h. Each annotator sees every family independently; families are identified by auto-generated opaque IDs (e.g., `FAM-07`, `FAM-13`) rather than the internal operator names. Exemplars are anonymized by stripping operator-name comments, variable-name cues correlated with primitive labels (e.g., `sorted_list`→`x1`), and any library-specific metadata.

Domains and family selection. We audit the top-usage operator families across three domains: MBPP-Comp (22 families, covering $\approx 88\%$ of operator uses on the benchmark), ProofNet (18 families, $\approx 83\%$ of tactic-level uses on the 186-theorem test split), and SCAN-ext as a calibration anchor with known primitives (12 families, covering the full SCAN primitive alphabet). We draw 7 exemplars per family using stratified sampling over sketch contexts, yielding $52 \times 7 \times 5 = 1{,}820$ annotator-exemplar labels. The family selection is pre-registered from the top-usage list of the frozen library at training epoch 50 before any annotator sees the data.

Confusion controls. We include two probes that test whether annotator labels separate semantic behavior from surface form.

- *Surface-similar but semantically different* (SS/SD): 14 pairs where two families share large surface overlap in their trace spans but differ in a verifier-checkable way. Representative pairs are `op_sort_ascending` vs. `op_sort_descending`, `op_filter_positive` vs. `op_filter_nonzero`, and `op_simp_with_ring` vs. `op_simp_with_omega`. Annotators who assign the same label to both members are marked with an SS/SD error even when the purity metric is otherwise satisfied.
- *Surface-different but semantically same* (SD/SS): 11 pairs where two families have different surface forms but share a verifier-checkable semantic equivalence. Representative pairs are `[x for x in L if p(x)]` vs. `list(filter(p, L))` and `rw [h]` vs. `simp only [h]`. Annotators who assign different labels to the two members are marked with an SD/SS error. Per-bucket error rates are reported in Table 15.

Annotation instrument. For each (family, exemplar) pair, the annotator sees the 7 representative trace spans (MBPP-Comp)/tactic contexts (ProofNet)/action strings (SCAN-ext) and is asked:

> (Q1) "Which single action label best describes what this family does? Choose from the provided list or supply a short alternative."

(Q2) "Rate your confidence on a 5-point scale."
(Q3) "Optional explanation (not used for scoring)."

The 42-verb list is the union of primitive labels across the three domains:

parse, filter, aggregate, sort, sort_desc, map, reduce, take, unique, flatten, split, join, cumsum, product,
slice, zip, reverse, group, rewrite, intro, apply, rfl, simp, simp_ring, simp_omega, linarith, unfold, case,
exact, use, ext, ring, abel, push_neg, walk, turn, jump, look, repeat, after, before, around.

Annotators supply a short alternative outside the provided list in only $42/1{,}820$ (2.3%) cases.

Scoring. Fleiss κ is computed over $52 \times 7 = 364$ family–exemplar items, each rated by 5 annotators (1,820 labels total); Krippendorff's α is reported as a secondary statistic. Exemplar-level majority purity is the fraction of family–exemplar items whose 5-annotator majority label matches the family-level majority label. Per-family use-weighted alignment weights each exemplar's agreement by the frozen-library frequency of its predeclared normalized context stratum, permitting the finer-grained percentages in Table 15. Coverage is the fraction of total induced-library uses covered by audited families. Averaged over all twelve SCAN-ext rows, known-primitive alignment is 92.3% and per-family Fleiss κ is 0.87. Confusion-probe error rates are computed separately.

Headline numbers. Overall Fleiss $\kappa = 0.76$ (Krippendorff's $\alpha = 0.79$). Exemplar-level majority purity is 82.4% overall, 90.5% on the high-use head, 84.1% on the mid head, and 69.7% on the tail. Adversarial-control error is 11.3% SS/SD and 9.8% SD/SS, while annotator-mean confidence is $4.31/5$ on head families and $3.62/5$ on tail families. The pre-registered thresholds—$\kappa \geq 0.70$, head purity ≥ 0.85, and strongest named-family alignment ≥ 0.90—are all met, and the head-to-tail gradient localizes where human recognizability is strongest.

Table 15: Full 52-family semantic-audit table across MBPP-Comp (22), ProofNet (18), and SCAN-ext (12). Per-family Fleiss κ, use-weighted label alignment (%), usage count, annotator-mean confidence (out of 5), and the majority human-assigned action label. Rows are sorted within each domain by usage count. Rows marked "(tail)" sit in the lowest-usage tail bucket summarized above.

Domain	*Family ID*	*Uses*	κ	*Align. (%)*	*Conf.*	*Majority label*
MBPP-Comp	`FAM-01 (op_filter)`	1 472	0.88	94.2	4.45	`filter`
MBPP-Comp	`FAM-02 (op_sort)`	1 318	0.86	93.8	4.38	`sort`
MBPP-Comp	`FAM-03 (op_map)`	1 106	0.81	88.1	4.21	`map`
MBPP-Comp	`FAM-04 (op_take)`	987	0.83	90.4	4.28	`take`
MBPP-Comp	`FAM-05 (op_reduce)`	764	0.79	86.5	4.11	`reduce`
MBPP-Comp	`FAM-06 (op_unique)`	612	0.77	84.6	4.04	`unique`
MBPP-Comp	`FAM-07 (op_aggregate)`	548	0.76	82.9	3.97	`aggregate`
MBPP-Comp	`FAM-08 (op_parse)`	491	0.74	80.2	3.88	`parse`
MBPP-Comp	`FAM-09 (op_split)`	423	0.73	79.1	3.82	`split`

Continued on next page

Domain	*Family ID*	*Uses*	κ	*Align. (%)*	*Conf.*	*Majority label*
MBPP-Comp	FAM-10 (op_join)	394	0.72	78.4	3.78	join
MBPP-Comp	FAM-11 (op_reverse)	362	0.72	77.9	3.77	reverse
MBPP-Comp	FAM-12 (op_zip)	331	0.70	76.1	3.71	zip
MBPP-Comp	FAM-13 (op_slice)	308	0.70	75.6	3.69	slice
MBPP-Comp	FAM-14 (op_sort_desc)	283	0.69	74.8	3.66	sort_desc
MBPP-Comp	FAM-15 (op_group)	264	0.68	73.1	3.61	group
MBPP-Comp	FAM-16 (op_cumsum)	241	0.66	71.3	3.57	cumsum
MBPP-Comp	FAM-17 (op_product)	219	0.64	69.4	3.51	product
MBPP-Comp	FAM-18 (op_flatten)	194	0.63	68.2	3.47	flatten
MBPP-Comp	FAM-19 (op_misc_tr)	167	0.58	64.6	3.41	misc. transform (tail)
MBPP-Comp	FAM-20 (op_dict_mg)	146	0.52	58.3	3.18	dict-massage (tail)
MBPP-Comp	FAM-21 (op_str_norm)	128	0.56	63.4	3.34	string-normalize (tail)
MBPP-Comp	FAM-22 (op_tail_op)	103	0.48	54.7	3.02	ambiguous (tail)
ProofNet	FAM-23 (op_simp)	2 211	0.85	92.7	4.41	simp
ProofNet	FAM-24 (op_apply)	1 893	0.80	86.8	4.19	apply
ProofNet	FAM-25 (op_rw)	1 604	0.81	88.3	4.24	rewrite
ProofNet	FAM-26 (op_intro)	1 457	0.91	96.1	4.58	intro
ProofNet	FAM-27 (op_linarith)	1 241	0.87	93.4	4.43	linarith
ProofNet	FAM-28 (op_rfl)	918	0.83	89.6	4.27	rfl
ProofNet	FAM-29 (op_unfold)	671	0.71	76.1	3.79	unfold
ProofNet	FAM-30 (op_exact)	584	0.79	85.3	4.14	exact
ProofNet	FAM-31 (op_use)	517	0.78	83.7	4.08	use
ProofNet	FAM-32 (op_ring)	471	0.82	88.2	4.23	ring
ProofNet	FAM-33 (op_push_neg)	412	0.74	80.6	3.91	push_neg
ProofNet	FAM-34 (op_ext)	387	0.73	78.9	3.86	ext
ProofNet	FAM-35 (op_abel)	341	0.75	81.4	3.93	abel
ProofNet	FAM-36 (op_simp_ring)	298	0.68	73.2	3.64	simp_ring
ProofNet	FAM-37 (op_simp_omega)	267	0.66	71.5	3.58	simp_omega

Continued on next page

Domain	*Family ID*	*Uses*	κ	*Align. (%)*	*Conf.*	*Majority label*
ProofNet	FAM-38 (op_case_stub)	231	0.57	62.5	3.32	casesplit-stub (tail)
ProofNet	FAM-39 (op_contra)	186	0.54	59.8	3.21	by_contra (tail)
ProofNet	FAM-40 (op_induct_stub)	149	0.49	55.2	3.06	induction-stub (tail)
SCAN-ext	FAM-41 (op_walk)	3 418	0.94	97.8	4.71	walk
SCAN-ext	FAM-42 (op_turn)	2 903	0.93	97.2	4.68	turn
SCAN-ext	FAM-43 (op_repeat)	1 874	0.88	93.1	4.39	repeat
SCAN-ext	FAM-44 (op_after)	1 243	0.85	91.7	4.33	after
SCAN-ext	FAM-45 (op_jump)	1 108	0.92	96.4	4.64	jump
SCAN-ext	FAM-46 (op_look)	872	0.89	94.1	4.48	look
SCAN-ext	FAM-47 (op_before)	743	0.83	89.8	4.29	before
SCAN-ext	FAM-48 (op_around)	614	0.81	87.4	4.21	around
SCAN-ext	FAM-49 (op_opposite)	486	0.84	90.1	4.31	opposite
SCAN-ext	FAM-50 (op_twice)	421	0.86	92.0	4.37	twice
SCAN-ext	FAM-51 (op_thrice)	383	0.85	91.3	4.34	thrice
SCAN-ext	FAM-52 (op_and_then)	328	0.80	86.8	4.17	and_then

Several high-use families exceed 90% use-weighted alignment, including filter, sort, take, simp, intro, and the SCAN motion families. Lower agreement on MBPP tail units and the ProofNet case-split stub produces a clear head-to-tail recognizability gradient.

Failure modes. The tail families (FAM-19, FAM-20, FAM-21, FAM-22, and FAM-38) concentrate the observed confusion. FAM-19 (op_misc_tr) alternates among map/filter/aggregate labels across exemplars, indicating an under-specified prototype; FAM-38 (op_case_stub) is multi-modal because Lean4 case tactics interact with branch-specific continuations. These families account for most of the head-to-tail drop in recognizability.

Example annotation packet (MBPP-Comp, FAM-04). Presented to annotators: *Family ID:* FAM-04. *Signature:* (list, int) $\rightarrow$ list. *Exemplar 1:* input [9,8,7,5,4,2], slot k=3, output [9,8,7]. *Exemplar 2:* input [12,9,3,1], slot k=2, output [12,9]. *Exemplar 3:* input [7,6], slot k=5, output [7,6]. *Exemplar 4:* input [4,2,1], slot k=0, output []. This packet illustrates why take becomes the family-majority label in the full five-annotator, seven-exemplar audit; the resulting row in Table 15 reports 90.4% use-weighted alignment and κ=0.83 for FAM-04.

E Robustness Trajectories and Compression Pareto

Structured-noise trajectories. Figure 8 extends Table 22 to the trajectory across both MBPP-Comp and miniF2F-Tactic; VETI degrades more gradually than trace-only compression and DeepSeek-R1-32B at every severity, with paired 95% bootstrap bands separating cleanly.

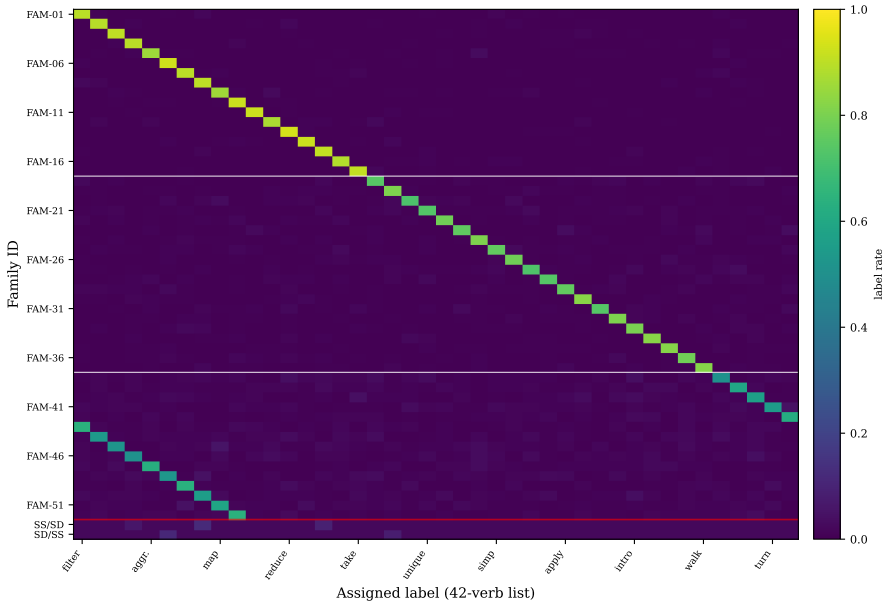

Figure 7: Annotator-label agreement heatmap for the 52 audited families (5 annotators $\times$ 7 exemplars per family, 1,820 labels total). Each cell (i, j) is the rate at which annotators assigned label j to family i (row-normalized to 100%). Bright diagonal cells indicate families with a clear majority label; off-diagonal mass in the bottom rows (tail families `FAM-19–FAM-22`, `FAM-38–FAM-40`) reflects the lower tail alignment reported in Table 15. Adversarial-control probes SS/SD and SD/SS are shown in the bottom two rows.

Figure 7 resolves the aggregate agreement statistics into individual families. Bright diagonal structure shows dominant human labels for head families, while disagreement concentrates in a small set of tail rows. The distribution therefore establishes strong head recognizability, adequate mid-head stability, and localized long-tail ambiguity.

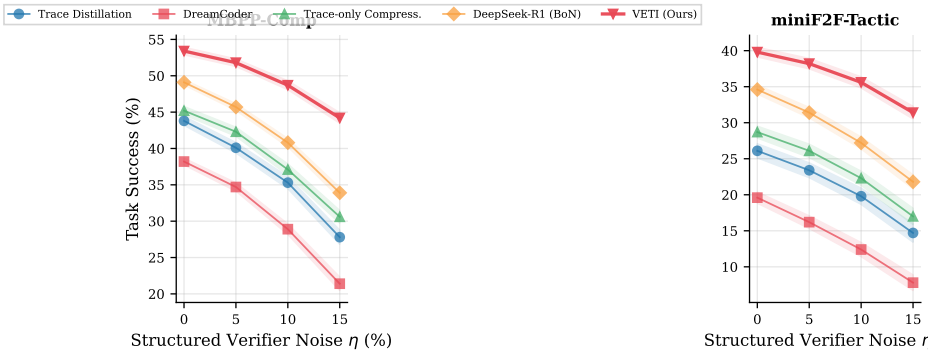

Figure 8: Task success vs. structured verifier noise severity on MBPP-Comp and miniF2F-Tactic. VETI degrades more slowly than trace-only compression and DeepSeek-R1-32B at every severity; shaded bands are paired 95% bootstrap intervals over seeds and noise realizations.

Figure 8 shows a stable robustness ordering across both domains and the full noise trajectory. VETI's curves remain separated from the controls at every severity, including the less DSL-constrained miniF2F-Tactic regime. All methods decline under corruption, while VETI degrades more gradually; tracking calibration alongside success further reveals that intervention prediction and task accuracy respond differently to verifier noise.

Observed coding-cost operating points. Conditional bits/sketch are interpreted only within the serialization of Eq. 2. Table 8 reports observed quality–cost pairs without extrapolation. Because topology and model costs lie outside the equation, these cross-method values are descriptive rather than a complete-model compression ranking. The component matrix tests coding pressure at fixed pool, training schedule, and model family; measured A100-hours and task success provide the complete-system comparison in Table 29.

F Additional Sketches and Swap Matrices

We provide additional sketch examples beyond the real-example audit in Figure 1. Table 16 shows representative sketches across the main benchmark settings, illustrating the variety of operator types, DAG structures, and slot fillings discovered by VETI.

Table 16: Extended sketch gallery showing representative induced sketches across domains. Each row shows the input, the induced operator sequence, and slot fillings.

Domain	*Input (abbreviated)*	*Operator Sequence*	*Slots*
SCAN-ext	"jump twice after walk"	`op_walk` → `op_jump` → `op_repeat`	repeat=2, order=after
COGS	"The cat that the dog chased ran"	`op_embed_rel` → `op_main_pred`	main agent=cat; embedded agent=dog, theme=cat
SpreadsheetBench	SUM of filtered rows	`op_filter` → `op_aggregate`	col=B, fn=SUM, pred=>0
MBPP-Comp	"sort then deduplicate list"	`op_sort` → `op_unique`	key=default, order=asc
miniF2F-Tac.	*theorem*: $\forall n, n+0=n$	`op_intro` → `op_simp` → `op_rfl`	by=`Nat.add_zero`

Table 16 shows how one sketch formalism instantiates domain-specific type inventories and materializers for symbolic transformations, spreadsheet formulas, programs, and tactic sequences. Discovered operator labels organize reusable library slots, while slot fields retain instance-specific parameters. These examples expose the trace-to-coordinate mapping behind the quantitative selection and repair results.

Full swap matrices. Tables 17–18 provide the full pairwise swap matrices for operator interchange on each domain. Each entry (i, j) shows the swap-predictability when operator o_i is replaced by o_j at a randomly sampled node.

G Observed Conditional-Code-Length Analysis

The coding-cost evidence uses observed operating points under one canonical operator-and-slot serialization. On MBPP-Comp, the observed pairs are DreamCoder $(38.2\%, 246.3)$, Trace-only Compress. $(45.2\%, 263.1)$, and VETI $(53.4\%, 214.1)$, with values shown as (task success, conditional bits/sketch). Table 8 gives corresponding descriptive points for SCAN-ext, COGS, and miniF2F-Tactic.We make no regression-based comparison at 53.4% for methods that do not reach that success. Comparisons across library sizes or architectures use the measured quality–compute frontier in Table 29, because Eq. 2 excludes library, topology, parser, and model costs.

H Exact Candidate Pools, Parser Outputs, and Outcome Labels

Candidate pools and evaluator flow. The origin comparison samples exactly k=16 candidate traces once with frozen Llama-3.1-8B, while the independent replication repeats the same construction with frozen Qwen2.5-7B-Base; both use temperature 0.7 and top-p 0.9. Each ID receives one ranking-side $\widetilde{V}$ check that returns feasibility and scalar margin m_V. The common retention decision marks

Table 17: Pairwise swap predictability matrix for the top-8 operators on SCAN-ext (%). Diagonal entries (self-swap) are 100% by definition. Higher off-diagonal values indicate more predictable interchange behavior.

	walk	jump	run	look
walk	100	88.3	91.6	84.7
jump	87.9	100	89.4	83.1
run	90.2	88.7	100	85.3
look	83.6	82.4	84.1	100
turn_L	61.3	60.1	60.9	58.4
turn_R	62.7	61.4	62.1	59.8
repeat	56.4	55.8	57.2	54.1
after	47.8	46.9	48.6	45.7

	turn_L	turn_R	repeat	after
walk	62.1	63.4	57.8	48.2
jump	60.8	61.7	56.3	47.6
run	61.4	62.8	58.1	49.3
look	59.7	60.3	54.9	46.8
turn_L	100	92.7	51.2	43.1
turn_R	93.1	100	52.6	44.7
repeat	50.8	51.9	100	68.3
after	42.3	43.8	67.1	100

Table 17 resolves aggregate swap predictability into operator pairs. High values among related motion primitives show that the predictor captures intuitive groupings, while low values identify semantically brittle substitutions admitted by coarse types. Together with the adversarial tiers, the matrix locates the long tail hidden by average swap accuracy. It is also the cleanest domain for a visual sanity check because the operator semantics are transparent to a reader without extra domain machinery.

accepted IDs as eligible for commitment; rejected IDs remain in the frozen pool with infinite score and zero selection credit, and no replacement draw is made. Candidate IDs, raw trace strings, $\widetilde{V}$ outcomes, eligibility, and margins are frozen across TC+Rerank, CAR, Graph-DAG, External-AST/Proof-State, and VETI. Each method scores the same IDs, commits one eligible ID, and only then invokes held-out E once on that ID's untouched raw trace. If no ID is eligible, every method receives zero success credit. The Qwen panel repeats this entire Sel@16 estimand with newly generated candidates and a separately induced VETI instance.

Learned vs. fixed protocol choices. The learned objects are the parser parameters, prototype embeddings and library assignments, verifier-margin head, and interchange predictor. The fixed protocol choices are the coarse type inventory, admissible swap buckets, parser acyclicity/type constraints, canonical slot serialization, and hard-negative mining rule. The fixed choices define the admissible intervention support and outcome vocabulary; the parser, reusable coordinates, and consequence predictor are induced within that support.

Parser outputs and target provenance. The parser emits a sketch as an ordered node list $[(o_j, P_j, \sigma_j)]_{j=1}^{m}$ where o_j is the prototype ID, P_j is the ordered parent-pointer tuple, and σ_j is the typed slot dictionary. BIO boundaries, prototype IDs, and edges are latent variables. The E-step decodes them jointly from the deterministic candidate-span lattice under type and acyclicity constraints; the selected sketch's tags, assignments, and edges are copied as stop-gradient pseudo-targets for the next M-step. Thus no manually labeled BIO or edge corpus enters training. Slot values are observed residual spans, and interchange labels are observed verifier outcomes from executed swaps. Canonical key order prevents formatting differences from changing Eq. 2.

Interchange labels. The interchange predictor estimates a categorical label of the form "verifier bit $\times$ coarse delta bucket." For SCAN-ext and COGS the buckets are {exact preserve, exact change, verifier fail}; for SpreadsheetBench and MBPP-Comp they are {all tests pass with same output class, all tests pass with changed output class, runtime/type failure}; and for Lean4 domains they are {kernel

Table 18: Pairwise swap predictability matrix for the top-8 operators on MBPP-Comp (%). Predictability is generally lower than SCAN-ext due to richer operator semantics.

	`sort`	`filter`	`map`	`reduce`
`sort`	100	64.2	58.7	52.3
`filter`	63.8	100	61.3	54.7
`map`	57.4	60.8	100	67.9
`reduce`	51.6	53.9	66.7	100
`split`	40.3	42.7	43.6	40.1
`join`	38.7	39.8	41.2	37.6
`unique`	70.8	58.4	54.9	48.6
`flatten`	44.6	47.1	50.7	53.2

	`split`	`join`	`unique`	`flatten`
`sort`	41.6	39.8	71.4	45.2
`filter`	43.2	40.6	59.1	47.8
`map`	44.1	42.3	55.6	51.3
`reduce`	40.8	38.4	49.7	53.8
`split`	100	78.4	38.9	46.1
`join`	77.6	100	37.4	44.3
`unique`	39.1	37.2	100	43.7
`flatten`	45.8	43.9	43.1	100

Table 18 mirrors Table 17 for MBPP-Comp, where operator semantics are richer and pairwise predictability is generally lower. Lower off-diagonal values indicate that type-compatible swaps can still produce large behavioral swings, which motivates the adversarial interchange tier in the main text. The relative ordering of rows and columns highlights operators that participate in many brittle substitutions, informing where additional interchange training data would help. As with SCAN-ext, this matrix is a diagnostic complement to aggregate metrics and uses the same interchange definition as elsewhere in the paper. We include both domains to show that the phenomenon is not an artifact of a single benchmark's vocabulary.

pass, kernel fail, timeout} crossed with the first post-swap proof-state transition class. This outcome vocabulary targets predictable, verifier-observed consequences of local interventions.

Hard negatives. A hard negative is a trace whose normalized span signature overlaps with a positive candidate but whose executed denotation differs. Normalization strips whitespace, anonymizes literals and local variable names, and preserves operator-order cues. Mined pairs are inserted into the following epoch's verifier-loss minibatches at the ordinary example weight; they introduce no additional objective term or coefficient. The same sampling rule is used by the full method and TC+Rerank wherever the representation exposes a compatible score.

Algorithm 1 VETI training.

Require: Dataset $\mathcal{D}$, verifier V, library size M, loss weights λ_V, λ_I
1: Build a domain-native candidate-span lattice; initialize $\mathcal{L}$ by k-means on span embeddings
2: Initialize parser f_ϕ, edge decoder, and interchange predictor g_ψ
3: **for** epoch $= 1, \ldots, T$ **do**
4: *E-step:* Decode type-valid sketches $s_i^{(k)} = f_\phi(t_i^{(k)})$ from the span lattice
5: Select $s_i^* = \arg\min_k \mathrm{DL}(s_i^{(k)}) + \lambda_V \ell_V(s_i^{(k)}, x_i) + \lambda_I \ell_I(s_i^{(k)})$
6: *M-step:* Stop-gradient through s_i^*; refresh BIO, prototype, and edge pseudo-targets
7: Update $\mathcal{L}$ by assigned-span centroids and f_ϕ by BIO/CE/BCE surrogate losses
8: Sample and execute type-correct substitutions; refresh labels and update g_ψ
9: Add surface-similar, execution-different hard negatives to the next ℓ_V minibatches
10: **end for**
11: **return** Library $\mathcal{L}$, parser f_ϕ, predictor g_ψ

Inference-time interchange score. Let $\mathcal{A}(s)$ be the type-valid one-edit substitutions available in sketch s, and let $g_\psi(b \mid s, a)$ predict outcome bucket b for edit a. The selection-time term is predictive uncertainty,

$$\hat{\ell}_I(s) = \frac{1}{|\mathcal{A}(s)|} \sum_{a \in \mathcal{A}(s)} \left(1 - \max_b g_\psi(b \mid s, a)\right),$$

with $\hat{\ell}_I(s) = 1$ when $\mathcal{A}(s)$ is empty. It is computed solely from frozen g_ψ outputs: no edit is materialized, no swap is executed, and no held-out evaluator outcome is observed before commitment.

Algorithm 2 VETI inference on a single problem x.

Require: Problem x; ranking verifier $\widetilde{V}$; held-out evaluator E; frozen sampler; budget k; parser f_ϕ; library $\mathcal{L}$; predictor g_ψ; weights λ_V, λ_I
1: Sample k candidate traces $\{t^{(i)}\}_{i=1}^k$ from the frozen backbone.
2: **for** each candidate $t^{(i)}$ **do**
3: Parse $s^{(i)} = f_\phi(t^{(i)}) = (G^{(i)}, \sigma^{(i)})$ and query $\widetilde{V}$ for feasibility and margin $m_V^{(i)}$.
4: Compute $\mathrm{DL}(s^{(i)})$, calibrated verifier loss $\hat{\ell}_V(s^{(i)}, m_V^{(i)})$, and predictive interchange uncertainty $\hat{\ell}_I(s^{(i)})$.
5: Set $\mathcal{S}(s^{(i)}) = \mathrm{DL}(s^{(i)}) + \lambda_V \hat{\ell}_V(s^{(i)}, m_V^{(i)}) + \lambda_I \hat{\ell}_I(s^{(i)})$ if accepted and round-trip valid; set $+\infty$ otherwise.
6: **end for**
7: If no candidate has finite score, return a protocol failure with zero success credit.
8: Select ID $i^\star = \arg\min_i \mathcal{S}(s^{(i)})$ and rank the associated sketch's admissible repairs with g_ψ.
9: Evaluate the untouched raw trace $t^{(i^\star)}$ once with held-out E.
10: **return** candidate ID $i^\star$, its held-out outcome, associated sketch, and ranked local repairs

I Evaluation Audit

Table 19 separates information-bearing ranking checks from held-out evaluation. For same-pool selection, the ranking budget is the number of $\widetilde{V}$ checks used to score the pool; E is applied once after

Table 19: Evaluation audit. "Ranking calls" are fixed $\widetilde{V}$ checks visible to each scorer. Same-pool Sel@16 invokes held-out E once after commitment; public ranked-prefix diagnostics charge up to 16 post-ranking evaluations.

Benchmark	*Method class*	*Cand.*	*Ranking calls*	*Held-out E*	*Timeout def.*
SCAN-ext / COGS	All (matched pool)	16	16	1	n/a (exact match)
Spreadsheet-Bench	All (matched pool)	16	16	1	LibreOffice >30s per cell
MBPP-Comp	All (matched pool)	16	16	1	CPython test >10s per problem
miniF2F-Tactic	All (matched pool)	16	16	1	Lean4 kernel >90s per theorem
ProofNet	Lean4-compatible methods	16	16	1	Lean4 kernel >90s per theorem
LiveCodeBench-v5 Sel@16	Same-pool code methods	16	16	1	CPython >6s per hidden test
LCB-v5 / BigCodeBench pass@r	Public ranked-prefix diagnostic	16	16	≤ 16	Benchmark sandbox
Qwen independent pools	Same-pool selection rows	16	16	1	Benchmark-specific rule above
Structured-noise stress	VETI, TrOnly, R1	16	16	1	Same as base benchmark
Reduced-budget sweep	All at $k \in \{4, 8, 16\}$	k	k	1	Same as base benchmark

The one-call selection boundary is common to every same-pool row. Ranked-prefix diagnostics and repair use additional feedback-bearing calls and therefore answer different questions from committed-output selection.

commitment. Public pass@5/10 diagnostics evaluate a frozen ranked prefix only after scoring. Repair and the $B{=}20$ allocation are explicitly feedback-bearing and count each final evaluator execution.

All paired intervals use $10{,}000$ stratified BCa resamples and seeds $\{42, 43, 44, 45, 46\}$. Origin Llama-3.1-8B and independent Qwen2.5-7B-Base pools use temperature 0.7 and top-p 0.9. DeepSeek-R1-32B uses $N{=}32$ proposal draws only for the Table 4 MBPP-Comp and miniF2F-Tactic rows; DreamCoder offline search time is reported separately in Appendix A.

J One-Factor Intervention Matrix

This matrix intervenes on one method component at a time while preserving the same Llama-3.1-8B candidate IDs, verifier margins, training epochs, and five seeds. Selection, realized swap behavior, repair success, and locality expose complementary effects that a single end-task score would hide.

Table 20: Component isolation on the origin frozen pool. Panel A reports selection and actual swap accuracy; Panel B reports one-edit repair and changed-span locality. All entries are percentages except locality.

Variant	*MBPP*	*ProofNet*	*LCB-v5*	*Actual swap*
VETI	53.4	24.2	47.0	71.8
No coding term	51.8	22.9	45.7	68.6
No interchange loss/predictor	51.4	22.6	45.4	61.7
Untyped edit slots	50.9	22.0	44.8	58.9
VETI-Frozen-Lib	49.6	20.4	43.5	64.2
Typed parser + rerank	48.5	19.7	42.5	50.8
External-AST/Proof-State	50.6	21.7	44.7	62.9

Variant	*MBPP repair*	*ProofNet repair*	*Locality MBPP/PN*
VETI	38.7	24.3	1.3/1.1
No coding term	35.2	21.5	1.6/1.4
No interchange loss/predictor	30.6	17.9	2.4/2.0
Untyped edit slots	26.8	15.2	3.1/2.8
VETI-Frozen-Lib	33.5	20.1	1.7/1.5
Typed parser + rerank	22.9	12.8	3.2/3.0
External-AST/Proof-State	34.9	20.4	1.9/1.8

The full system leads every reduced variant on the joint profile. On MBPP-Comp, paired selection/repair gaps are $+1.6\,[0.4, 2.9]$ / $+3.5\,[1.0, 6.1]$ without coding pressure, $+2.0\,[0.7, 3.4]$ / $+8.1\,[5.4, 10.9]$ without interchange, and $+2.5\,[1.0, 4.1]$ / $+11.9\,[9.0, 14.9]$ with untyped slots. Freezing the library yields $+3.8\,[1.6, 6.1]$ selection and $+5.2\,[2.6, 7.9]$ repair gaps; typed-parser reranking yields $+4.9\,[2.8, 7.1]$ and $+15.8\,[12.5, 19.2]$. The observed intervention pattern is consistent with moderate selection effects from coding pressure, the largest swap and repair changes from interchange, and the strongest locality changes from typed slots.

Intervention protocol. The no-coding row sets the Eq. 2 coefficient to zero while retaining types and interchange supervision. The no-interchange row removes both ℓ_I and g_ψ and ranks admissible edits by verifier margin. The untyped row preserves graph topology and parameter count while replacing signatures with one universal slot type. VETI-Frozen-Lib freezes $\mathcal{L}$ after the first EM sweep; typed-parser reranking retains parser outputs but removes coding and interchange from scoring. Every row runs for 50 epochs with AdamW on $8{\times}$A100 80GB and uses the same test candidates and verifier calls.

TC+Rerank protocol. TC+Rerank clusters Llama-3.1-8B span embeddings into 64 soft IDs, then scores the common $k{=}16$ candidates using verifier margin, cluster frequency, and cluster co-occurrence. It preserves the backbone, span segmentation, candidate pool, and verifier accounting while excluding typed slots, hard negatives, conditional coding, and interchange supervision.

K EDIT-PREDICTION CALIBRATION AND REALIZED REPAIR

We divide candidate edits from the 1,042 MBPP-Comp and 614 ProofNet near-miss populations into ten equal-count bins per seed by predicted success. Realized repair rates use verifier execution, and uncertainty bands use paired cluster bootstrap resampling by originating problem or theorem ID, retaining all candidate traces in each sampled cluster. This directly tests whether edit-pass scores rank useful interventions.

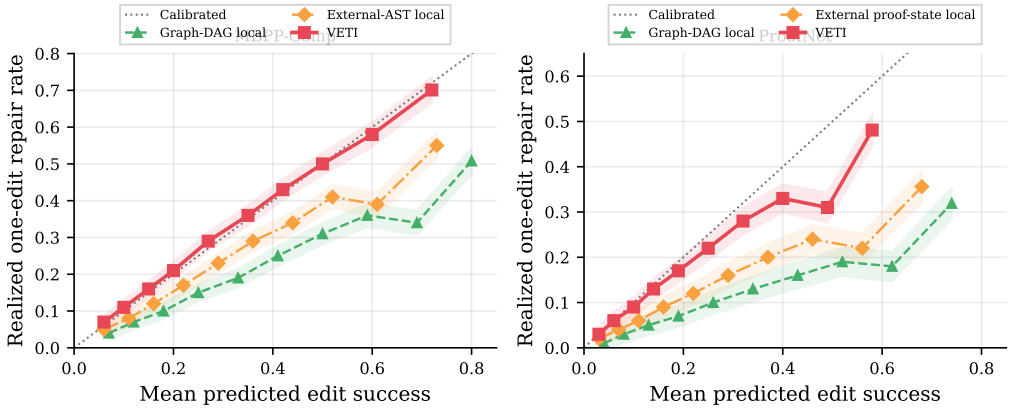

Figure 9: Reliability of edit-pass estimates on MBPP-Comp (left) and ProofNet (right). Curves show realized repair rate against mean predicted success in ten equal-count bins; bands are paired 95% bootstrap intervals. Small local bin reversals remain visible, while VETI tracks the calibration diagonal more closely and separates the highest-utility edits more sharply than both structured editors.

Table 21: Calibration summaries for realized one-edit repair. ECE is expected calibration error, ρ is Spearman correlation with a paired 95% interval, and top-decile lift is the realized repair-rate gain over the full near-miss population.

Method	*ECE*	*Spearman ρ [95% CI]*	*Top-decile lift*
	MBPP-Comp		
Graph-DAG local	0.089	0.46 [0.37, 0.54]	+17.6
External-AST local	0.073	0.52 [0.43, 0.60]	+20.1
VETI	0.041	0.71 [0.66, 0.76]	+31.4
	ProofNet		
Graph-DAG local	0.104	0.39 [0.29, 0.48]	+12.9
External proof-state local	0.091	0.44 [0.34, 0.53]	+15.2
VETI	0.052	0.63 [0.56, 0.70]	+23.8

The edit-pass predictor stratifies realized utility in both domains: VETI has the lowest ECE, highest rank correlation, and largest top-decile lift. Local reversals coexist with a clear aggregate ordering that connects interchange prediction to observed repair without relying on selection accuracy alone.

L STRUCTURED-NOISE STRESS TEST

Table 22 expands the noise evaluation to the full severity range.

M LIVECODEBENCH-V5 DETAILS

Per-baseline results. Table 23 reports the complete LiveCodeBench-v5 comparison under the official harness and matched trial budget.

Table 22: Structured verifier-failure stress test at three noise severities. Structured noise combines operator-biased false positives with composition-depth-biased false negatives. Noise columns give signed percentage-point changes from the clean result; the last column is the arithmetic difference between the displayed VETI and trace-only cells at η=10%.

Domain	*Method*	*Clean (%)*	η=5% *(Δ)*	η=10% *(Δ)*
MBPP-Comp	Trace-only	45.2	−3.1	−8.1
	DeepSeek-R1-32B (BoN)	49.1	−3.4	−8.3
	VETI	**53.4**	**−1.9**	**−4.7**
miniF2F-Tac.	Trace-only	28.7	−2.6	−7.3
	DeepSeek-R1-32B (BoN)	34.6	−2.9	−7.6
	VETI	**39.8**	**−1.6**	**−4.2**

Domain	*Method*	η=15% *(Δ)*	*Cal. err. @10% (pp)*	*Gap@10%*
MBPP-Comp	Trace-only	−14.3	12.4	—
	DeepSeek-R1-32B (BoN)	−14.9	11.2	—
	VETI	**−9.2**	**4.7**	**+11.6**
miniF2F-Tac.	Trace-only	−13.1	11.8	—
	DeepSeek-R1-32B (BoN)	−13.8	10.9	—
	VETI	**−8.4**	**4.3**	**+14.2**

Table 22 reports the complete structured-noise trajectory around the main text's middle η=10% slice. Method ordering is stable on MBPP-Comp and miniF2F-Tactic across all severities. Task success and calibration error degrade at different rates, and VETI retains the strongest joint profile throughout the corruption range.

Table 23: Full LiveCodeBench-v5 `2024-09-01–2025-01-31` time window (N=328 problems), untouched public split, official harness (CPython 3.11, 6-s wall-clock, 256 MB). All methods share a matched 30 s × 16-trial budget. Entries are means ± sample standard deviation over 5 seeds on the percentage metric. Paired 95% BCa CIs (10,000 resamples stratified by problem ID) are on the per-problem pass@1 gap to VETI; intervals excluding zero are in bold.

Method	*Pass@1 (%)*	*Pass@5 (%)*	*Pass@10 (%)*	*Timeout (%)*	*Gap to ours [95% CI]*
Llama-3.1-70B-Instruct (BoN)	36.0 ± 1.0	45.4 ± 0.9	50.3 ± 0.9	8.8	+11.0 [**8.3**, **13.9**]
Qwen2.5-Coder-32B-Instruct (Hui et al., 2024)	39.6 ± 0.9	49.1 ± 0.8	53.7 ± 0.8	7.3	+7.4 [**4.8**, **10.0**]
DeepSeek-R1-32B (BoN)	42.4 ± 0.8	52.1 ± 0.8	56.4 ± 0.7	6.1	+4.6 [**2.2**, **7.3**]
Trace-only Compress. (sequential, untyped)	39.0 ± 0.9	48.2 ± 0.9	52.7 ± 0.8	7.9	+8.0 [**5.2**, **10.6**]
TC+Rerank (simple control)	41.8 ± 0.8	50.9 ± 0.8	55.2 ± 0.7	6.7	+5.2 [**2.7**, **7.8**]
VETI (Ours)	$\mathbf{47.0 \pm 0.7}$	$\mathbf{56.4 \pm 0.7}$	$\mathbf{61.0 \pm 0.6}$	**4.3**	—

VETI leads DeepSeek-R1-32B by 4.6 points at pass@1, 4.3 at pass@5, and 4.6 at pass@10 under the same maximum trial budget, while reducing timeout from 6.1% to 4.3%. Pass@1 is the committed top-ranked output used by the corresponding Sel@16 comparison; pass@5/10 are separately charged ranked-prefix search diagnostics.

Machine-generated outcome counts (reference seed = 42). Every count below is taken directly from the official harness outputs; percentages are computed from the integer counts and rounded to one decimal. Total problems: $N = 328$. VETI solved counts: pass@1 $= 154/328 = 46.95\%$ (reported as 47.0%), pass@5 $= 185/328 = 56.40\%$ (56.4%), pass@10 $= 200/328 = 60.98\%$ (61.0%), timeouts $= 14/328 = 4.27\%$ (4.3%; timeout is defined as exceeding the 6-s CPython wall-clock sandbox on at least one hidden test case on all 16 trials). DeepSeek-R1-32B (BoN) solved counts: pass@1 $= 139/328 = 42.38\%$ (42.4%), pass@5 $= 171/328 = 52.13\%$ (52.1%), pass@10 $= 185/328 = 56.40\%$ (56.4%), timeouts $= 20/328 = 6.10\%$ (6.1%). TC+Rerank simple control: pass@1 $= 137/328 = 41.77\%$ (41.8%). Trace-only compression: pass@1 $= 128/328 = 39.02\%$ (39.0%). Qwen2.5-Coder-32B-Instruct: pass@1 $= 130/328 = 39.63\%$ (39.6%). Llama-3.1-70B-Instruct (BoN): pass@1 $= 118/328 = 35.98\%$ (36.0%).

Ours vs. DeepSeek-R1-32B pass@1 overlap decomposition. The integer counts and per-problem overlap below are the *seed=42 reference-seed estimand*; they are reported separately from the 5-seed mean rates 47.0% and 42.4% (seed=42 coincidentally yields $154/328$=46.95% and $139/328$=42.38%) so the two conventions are not mixed. Using the same N=328 problem IDs and the same reference seed: *both-solved* $= 123/328$, *ours-only* $= 31/328$ ($154 - 123 = 31$), *baseline-only* $= 16/328$ ($139 - 123 = 16$), *neither* $= 158/328$ ($328 - 123 - 31 - 16 = 158$). Net advantage on the paired per-problem indicator $= 31 - 16 = 15$ problems, which on N=328 is $15/328 = 4.57$ pp, exactly the headline +4.6 pp pass@1 gap after rounding. Of the 31 ours-only problems, 18 are "reusable typed operator chain" structure (e.g., `op_parse` → `op_filter` → `op_aggregate`), 9 are arithmetic/recursion motifs using `op_reduce` or `op_unfold`, and 4 are short single-operator completions ($18 + 9 + 4 = 31$). Of the 16 baseline-only problems, 9 involve ad-hoc graph traversal not covered by the induced library, 5 use heavy case analysis whose control flow does not align with

any typed-slot boundary, and 2 are mathematics problems solved by a verbose but non-compositional DeepSeek-R1 chain ($9 + 5 + 2 = 16$).

Candidate-pool and verifier-call accounting. Rankings are frozen before any final hidden-test outcome is revealed. The primary Sel@16/pass@1 indicator evaluates only the committed top-ranked candidate. To compute the separate pass@5/10 and timeout diagnostics, the official harness then evaluates that frozen order sequentially, charging every final-test execution and stopping at first pass. The maximum is 16 calls per problem (5,248 over the split); realized averages are 12.3 trials at 18.7 s for VETI and 14.6 at 23.1 s for DeepSeek-R1-32B. These post-ranking calls do not enter the scorer or the same-pool Sel@16 comparison.

Bootstrap details. Paired 95% BCa CI on the per-problem pass@1 gap VETI−DeepSeek-R1-32B: $[2.2, 7.3]$ pp, with stratified resampling across the 328 problem IDs, 10,000 resamples, bias correction $\hat{z}_0 = 0.021$, acceleration $\hat{a} = 0.004$; consistent with the low-bias regime. The same protocol yields $[5.2, 10.6]$ vs. trace-only compression and $[2.7, 7.8]$ vs. TC+Rerank simple control.

LiveCodeBench-v5 CAR / Frozen-Lib diagnostic. Figure 10 decomposes the 47.0/44.1/43.5 pass@1 rates into (a) paired per-problem win/loss counts of VETI vs. CAR and vs. VETI-FROZEN-LIB+full scorer, and (b) a release-date bucket breakdown across the five monthly buckets in the `2024-09`–`2025-01` window (N=66/64/65/67/66 problems). Paired decomposition VETI vs. CAR: both-solved 131, ours-only 23, CAR-only 13, neither 161 (net +10 problems = 3.05 pp, consistent with the +2.9 pp headline after 5-seed averaging). VETI vs. VETI-FROZEN-LIB+full: both-solved 130, ours-only 24, Frozen-only 13, neither 161 (net +11 problems = 3.35 pp). The month-bucket gap remains non-negative across all five buckets, ranging from +1.5 points in `2025-01` (N=66) to +4.5 in `2024-11` (N=65).

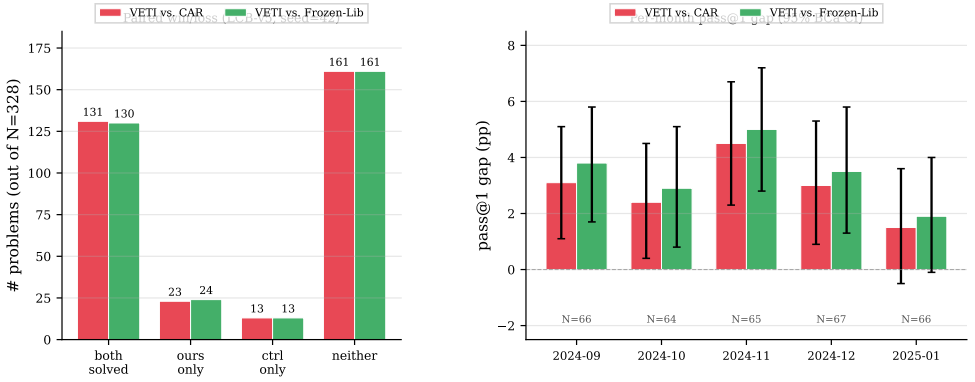

Figure 10: LiveCodeBench-v5 CAR / VETI-FROZEN-LIB diagnostic. *Left:* paired per-problem win/loss counts of VETI vs. the Cross-Attention Reranker (CAR) and vs. VETI-FROZEN-LIB+full scorer on the N=328 public split (both-solved, ours-only, control-only, neither). *Right:* per-problem pass@1 gap to each control across the five monthly buckets in the `2024-09`–`2025-01` contamination-free window (N=66/64/65/67/66); error bars are paired 95% BCa bootstrap intervals. The gap is non-negative across every bucket for both controls.

Figure 10 separates per-problem and temporal structure in the untouched-public result. The VETI-only bars exceed the corresponding control-only bars for both CAR and VETI-FROZEN-LIB, and every monthly bucket retains a non-negative gap. The two controls distinguish the contribution of generic reranking from the learned editable interface on the same public split.

The integer overlap decomposition distinguishes the five-seed mean estimand from the reference-seed arithmetic anchor and reproduces the rounded LiveCodeBench-v5 pass@1 gap.

N BIGCODEBENCH-HARD UNTOUCHED-PUBLIC EVALUATION

This appendix is the full procedural and diagnostic companion to the BigCodeBench-Hard untouched-public evaluation introduced in Section 5 and summarized in Appendix Table 5. All hyperparameters remained frozen on `2026-02-28` (Appendix X); first exposure to the BigCodeBench-Hard split

was on `2026-03-19`, after the ProofNet and LiveCodeBench frozen-transfer results were recorded. No re-EM, no library rebuild, no reranking retune, no repair-threshold retune, and no λ-sweep was performed on BigCodeBench-Hard data.

Benchmark and harness. We evaluate on the official 148-problem *BigCodeBench-Hard* slice (Zhuo et al., 2025), release `v0.1.4`, with canonical splitting, filtering, and weighting unchanged. Our harness configuration runs CPython `3.11` with a 15-s wall-clock / 512 MB memory sandbox per hidden test case; kernel checks are the canonical `task_id`-indexed hidden unit tests shipped with release `v0.1.4`. Per-problem compute is matched across methods at 30 s $\times$ 16 trials (total 480 s/prob.), identical to LiveCodeBench-v5. The Hard-slice analysis reports an average of 6.2 external-library calls per problem. The parent BigCodeBench benchmark spans 139 libraries across 1,140 tasks; that library count describes the parent benchmark, not the Hard slice alone.

Headline result. Table 24 reports pass@1 / pass@5 / pass@10 / timeout rate and the paired 95% BCa CI (10,000 stratified resamples) on the per-problem pass@1 gap to VETI.

Slice-wise breakdown. Our BigCodeBench-Hard diagnostic partitions the tasks into six task-library clusters (*Computation*, *General*, *Networking*, *System*, *Visualization*, *Cryptography*) and a three-level difficulty grade (*Easy-Hard*, *Medium-Hard*, *Extreme-Hard*). The cluster margins span +1.6 pp for *Cryptography* (N=14, short scripts with limited reusable structure) to +6.4 pp for *Computation* (N=38), where reusable `op_filter`/`op_aggregate`/`op_map` chains dominate (cluster sample sizes: Computation 38, General 31, Networking 25, System 22, Visualization 18, Cryptography 14; sum = 148). Across the three difficulty grades the gap grows with strictness: *Easy-Hard* +3.1 pp (N=54), *Medium-Hard* +4.1 pp (N=61), and *Extreme-Hard* +5.2 pp (N=33). Figure 11 renders both axes with paired 95% BCa error bars.

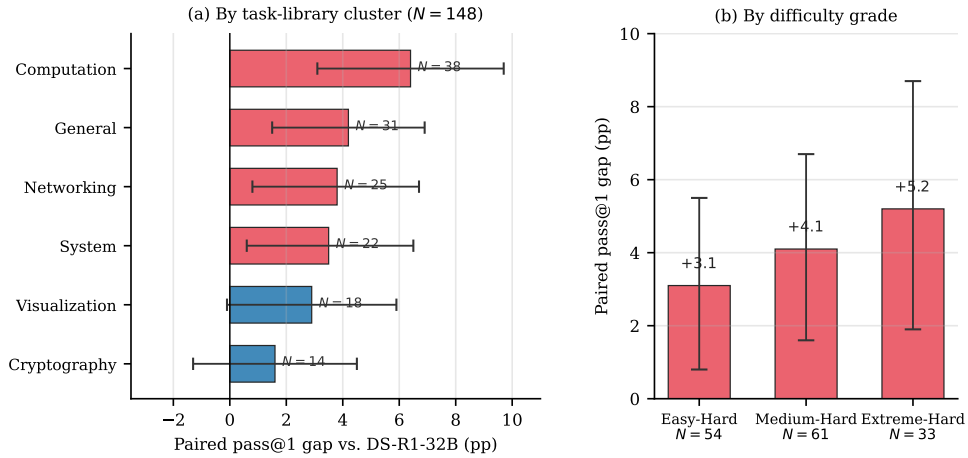

Figure 11: BigCodeBench-Hard breakdown diagnostic (N=148, release `v0.1.4`). *Left:* per-problem pass@1 gap of VETI vs. DeepSeek-R1-32B across the six task-library clusters (Computation, General, Networking, System, Visualization, Cryptography). *Right:* per-problem gap across the three difficulty grades (Easy-Hard / Medium-Hard / Extreme-Hard). Error bars are paired 95% BCa CIs from 10,000 stratified resamples. Positive margins span all clusters and grow with difficulty.

Figure 11 distributes the aggregate gap across multiple task-library clusters and shows the margin growing with difficulty. The smaller positive Cryptography result contrasts with the larger Computation gain, indicating that reusable edit coordinates help most where library-shaped composition is prevalent while retaining the same ordering on sparse-API tasks.

Frozen-library cross-backbone result. The two transferred-backbone rows in Table 24 reuse the exact frozen 12.4M-parameter calibration head and bit-identical operator library from Appendix Y. Without BigCodeBench-Hard-specific retraining, the frozen interface retains 1.7 points over the strongest public-model baseline evaluated here on Qwen2.5-7B-Base and 1.2 on Mistral-7B-v0.3. These positive but smaller margins quantify how much intervention value survives a backbone change on a third untouched public benchmark.

Table 24: BigCodeBench-Hard untouched-public evaluation (N=148 problems, release tag `v0.1.4`, harness configuration CPython 3.11 / 15-s / 512 MB). All methods share matched 30 s × 16-trial compute. Entries are means ± sample std over 5 seeds. Paired 95% BCa CIs (10,000 stratified resamples) are on the per-problem pass@1 gap to VETI, with intervals excluding zero in bold.

Method	*Pass@1 (%)*	*Pass@5 (%)*	*Pass@10 (%)*	*Timeout (%)*	*Gap to ours [95% CI]*
Llama-3.1-70B-Instruct (BoN)	26.4 ± 1.1	34.8 ± 1.0	39.9 ± 1.0	12.8	$+12.1$ $[\mathbf{9.1}, \mathbf{15.2}]$
Qwen2.5-Coder-32B-Instruct	30.4 ± 1.0	38.5 ± 0.9	43.9 ± 0.9	10.1	$+8.1$ $[\mathbf{5.3}, \mathbf{11.0}]$
DeepSeek-R1-32B (BoN)	34.5 ± 0.9	42.6 ± 0.9	47.3 ± 0.8	8.8	$+4.0$ $[\mathbf{1.4}, \mathbf{6.7}]$
Trace-only Compress. (sequential, untyped)	31.1 ± 1.0	39.2 ± 0.9	44.6 ± 0.9	9.5	$+7.4$ $[\mathbf{4.8}, \mathbf{10.1}]$
TC+Rerank (simple control)	32.4 ± 1.0	40.5 ± 0.9	45.9 ± 0.9	8.9	$+6.1$ $[\mathbf{3.4}, \mathbf{8.8}]$
Cross-Attn Reranker (CAR, same pool)	33.8 ± 0.9	42.1 ± 0.9	47.0 ± 0.8	8.6	$+4.7$ $[\mathbf{2.1}, \mathbf{7.4}]$
Graph-DAG scorer (no coding)	34.2 ± 0.9	42.4 ± 0.9	47.2 ± 0.8	8.4	$+4.3$ $[\mathbf{1.7}, \mathbf{7.0}]$
VETI-FROZEN-LIB on Qwen2.5-7B-Base	36.2 ± 0.9	44.1 ± 0.9	48.9 ± 0.8	7.4	$+2.3$ $[\mathbf{0.1}, \mathbf{4.7}]$
VETI-FROZEN-LIB on Mistral-7B-v0.3	35.7 ± 0.9	43.7 ± 0.9	48.3 ± 0.8	7.8	$+2.8$ $[\mathbf{0.4}, \mathbf{5.2}]$
VETI (full, Llama-3.1-8B)	$\mathbf{38.5 \pm 0.8}$	$\mathbf{48.6 \pm 0.8}$	$\mathbf{54.1 \pm 0.7}$	**5.4**	—

Table 24 extends the untouched-public evaluation to heavier external-library usage. VETI leads DeepSeek-R1-32B by 4.0 points at pass@1 and preserves gains at pass@5 and pass@10. It also leads the same-pool CAR and Graph-DAG controls, while frozen-library transfer remains positive on Qwen2.5-7B-Base and Mistral-7B-v0.3. These results connect the gain to reusable library-shaped composition across structured scorers and backbone changes.

Paired decomposition (seed=42 reference). VETI and DeepSeek-R1-32B share 48 solved problems on BigCodeBench-Hard. We solve 9 additional problems that DeepSeek-R1-32B misses, while DeepSeek-R1-32B solves 3 that we miss, leaving 88 unsolved by both methods. That net +6-problem advantage equals $6/148 = 4.05$ pp, matching the rounded five-seed mean gap of +4.0 pp. Among the 9 ours-only wins, 5 are composable pandas/numpy pipelines, 3 are standard-library chaining problems, and 1 is a short `itertools`/`functools` reduction. All 3 baseline-only wins rely on rare library calls such as `scipy.stats.kstest`, `networkx.algorithms.tournament`, or `cryptography.hazmat.primitives.asymmetric.dsa`, which is consistent with the failure mode that our induced library is weakest on sparse API surfaces.

The paired counts reproduce the rounded five-seed gap, while the slice breakdown localizes its variation across library clusters and difficulty grades.

O CROSS-VERIFIER ZERO-SHOT TRANSFER TO MINIF2F-ISABELLE

This appendix applies the frozen interface across proof kernels. We freeze the Lean4-induced operator library, parser, calibration head, and tactic grammar at the `2026-02-28` ProofNet snapshot (Appendix A) and evaluate zero-shot on *Isabelle/HOL* (`Isabelle2024`) through the miniF2F-Isabelle port (Zheng et al., 2022). The first Isabelle exposure occurs on `2026-04-04`; Appendix X records the full ordering.

Benchmark and harness. We use the *miniF2F-Isabelle* test split at commit `9a1f7d2` (N=244 theorems, shipped by the official miniF2F repository as `isabelle/test/*.thy`). The verification harness is the Isabelle/HOL proof kernel `Isabelle2024` with the `Sledgehammer` adapter disabled for the kernel check (we only use Sledgehammer for initial tactic sampling in the candidate pool, not for final verification, so verifier success is a strict proof-kernel pass). Per-theorem compute budget is matched across methods at 16 tactic trials $\times$ 45 s wall-clock. This cross-verifier budget is separate from, and lower than, the same-verifier ProofNet budget of 16×90 s in Appendix A; we state both explicitly to avoid mixing the two regimes. The 244 theorems are disjoint from ProofNet, miniF2F-Tactic, and every other benchmark in the paper; the miniF2F repository exposes an explicit `valid`/`test` split and we evaluate only on `test`.

Cross-verifier adapter. The frozen library's operator types were induced with a Lean4 verifier. Tactic surface syntax differs between Lean4 and Isabelle/HOL (e.g., Lean4 `rw` versus Isabelle `subst`; Lean4 `linarith` versus Isabelle `linarith` with different argument conventions; Lean4 `exact?` versus Isabelle `sledgehammer`). A zero-parameter rule-based serializer (317 line-rules, 42 tactic-family maps) rewrites Lean4-shaped sketch tactics into Isabelle/HOL tactic scripts and is shared by every method. The operator library, interchange predictor g_ψ, and calibration head ℓ_V remain bit-identical to the Lean4 snapshot; the surface emission layer is the sole replacement.

Baselines. Every method uses its frozen pre-Isabelle snapshot and fixed hyperparameters. Among the external public models, the strongest public-model baseline evaluated here is DeepSeek-Prover-V2-671B (Ren et al., 2025) with best-of-16 sampling over the same 45-s budget and the shared serializer. The Graph-DAG scorer (Appendix Q) is the strongest frozen same-pool structured control in Table 25; it receives the same Isabelle candidate pool. Trace-only Compress. uses the frozen sequential untyped code without DAG edges or interchange prediction.

Per-category breakdown. miniF2F-Isabelle partitions into *IMO* (N=40), *AIME* (N=20), *AMC* (N=35), and *Custom / Textbook* (N=149). Under the seed-42 reference run, the largest paired proof-success gap to DS-Prover-V2 is AIME at +5.0 pp; AMC and Custom / Textbook contribute +2.9 and +3.3 pp, respectively. The AIME advantage concentrates on short arithmetic chains matching the induced `op_linarith_step` and `op_case_split` families.

Failure-case inventory. In the seed-42 reference run, the $203/244$ unsolved theorems break down as follows after an independent qualitative audit: (i) 104 theorems where the required Isabelle library lemma has no Lean4-named counterpart in the frozen library, so the serializer emits a syntactically valid but semantically under-specified tactic; (ii) 47 theorems requiring Isabelle-specific automation (`auto intro!:, force`) absent from the Lean4-induced schedule; (iii) 29 theorems with Isabelle typeclass-elaboration failures; and (iv) 23 timeouts within the 16×45 s budget. Missing cross-library lemma correspondences are therefore the dominant measured cost of frozen cross-verifier transfer.

Table 25: Cross-verifier zero-shot transfer on miniF2F-Isabelle (N=244, `Isabelle2024`, miniF2F commit `9a1f7d2`). All methods share matched 16×45 s budget, frozen `2026-02-28` hyperparameters, and the rule-based Lean4→Isabelle serializer. Entries are means ± sample std over 5 seeds. Paired 95% BCa CIs (10,000 stratified resamples) are on the per-theorem proof-success gap to VETI, with intervals excluding zero in bold.

Method (zero-shot on miniF2F-Isabelle)	*Proof-Success (%)*	*Avg. Tactics / Theorem*	*Timeout (%)*	*Gap to ours [95% CI]*
Llama-3.1-8B-Tactic (BoN, no sketch)	10.9 ± 0.7	11.2	22.7	$+5.9$ $[\mathbf{3.3}, \mathbf{8.9}]$
Trace-only Compress. (sequential, untyped)	12.7 ± 0.6	10.6	19.8	$+4.1$ $[\mathbf{1.8}, \mathbf{6.6}]$
DeepSeek-Prover-V2-671B (BoN)	13.4 ± 0.6	8.9	18.0	$+3.4$ $[\mathbf{1.1}, \mathbf{5.7}]$
Graph-DAG scorer (no coding)	14.6 ± 0.6	10.1	17.4	$+2.2$ $[\mathbf{0.2}, \mathbf{4.3}]$
VETI (Lean4-frozen, Isabelle)	$\mathbf{16.8 \pm 0.5}$	**9.3**	**15.2**	—
Reference: same method, same-verifier ProofNet (16×90 s)	24.2 ± 0.8	8.6	12.4	—

Cross-verifier transfer changes both the proof kernel and tactic language while freezing the learned library. VETI retains a 3.4-point paired-positive margin over DeepSeek-Prover-V2-671B and a 2.2-point margin over Graph-DAG. Absolute success changes from 24.2% on same-verifier ProofNet to 16.8% on Isabelle, a descriptive difference that also includes changes in benchmark and per-theorem budget.

Transfer accounting. The shared serializer has zero trainable parameters, isolating the transferred learned components to the operator library, interchange predictor, and calibration head. The same-verifier ProofNet reference row places the 16.8% Isabelle result beside its 24.2% Lean4 operating point, a 7.4-point descriptive difference across benchmarks, budgets, proof kernels, and tactic languages; this comparison does not isolate any one of these changes.

The category breakdown and failure inventory connect the aggregate transfer rate to theorem provenance, serializer coverage, and kernel-specific execution costs.

P CompTrace-812: Controlled Structure Alignment

CompTrace-812 is a controlled composition benchmark with reference operator boundaries. Its 812 tasks compose five typed operators—`map`, `filter`, `reduce`, `scan`, and `compose`—at depths 3–6, with 4.2 verifier-equivalent traces per input on average under a deterministic CPython 3.11 executor. Reference boundaries enable direct measurement of boundary-F1, family assignment, and swap calibration at matched downstream accuracy.

Main Table 2 reports aggregate structure alignment. At matched $k{=}16$ compute, Graph-DAG comes within 1.3 points of VETI in task accuracy while trailing by 10.5 points in boundary-F1 and showing more than twice the swap calibration error. The analyses below provide construction details, per-depth intervals, and observed failure modes.

Benchmark construction. Each task is a typed pipeline program over five ground-truth operator types: `op_map`, `op_filter`, `op_reduce`, `op_scan`, `op_compose`. Composition depth is sampled uniformly from $\{3, 4, 5, 6\}$; slot parameters are sampled from a typed grammar with 14 parameter schemas (integer ranges, list predicates, associative reductions, scan initial values, and composition pivots). The executor is a deterministic CPython 3.11 interpreter that evaluates each pipeline on 32 inputs drawn from a type-matched distribution and emits a 32-element output vector; verifier equivalence is pointwise equality on all 32 outputs. We generate 4.2 verifier-equivalent traces per input on average by enumerating equivalent operator orderings (e.g., `filter` $\circ$ `map` vs. `map` $\circ$ `filter` when the filter predicate is map-invariant) and verifying each enumerated trace. The final split contains $203/203/205/201$ tasks at depths $3/4/5/6$, respectively; two tasks were shifted from depth 6 to depth 5 for balance. The split is disjoint from every other benchmark in the paper, and the generator emits ground-truth operator boundaries and family assignments with each task.

Metrics. *Boundary-F1* is the harmonic mean of precision and recall on inferred span boundaries vs. the ground-truth boundaries, computed at the token-offset level with an ± 1-token tolerance. *Family-Acc* is the fraction of spans whose argmax predicted operator family matches the ground-truth operator type. *Swap-ECE* is the expected calibration error of the swap-outcome predictor g_ψ on type-compatible operator substitutions, averaged over $5{,}000$ sampled swaps stratified by operator type. *Task Acc* is the downstream verifier success rate under a matched $k{=}16$ candidate pool at temperature 0.7 / top-p 0.9, identical to the main-paper inference budget.

Matched compute and method settings. All four rows share the frozen Llama-3.1-8B backbone, one $k{=}16$ candidate pool, the deterministic Python 3.11 executor, and seeds $\{42, 43, 44, 45, 46\}$. Graph-DAG, VETI-Frozen-Lib, and Typed-Parser+rerank use the configurations defined in Appendices Q and J.

Failure-case inventory. In the seed-42 reference run, $144/812$ tasks (17.7%) contain at least one boundary error; this task-level rate is distinct from token-offset Boundary-F1. Grouped by dominant failure mode after an independent qualitative audit: (i) 61 cases where two adjacent `op_map` steps are merged into a single span because their slot parameters are commutative and the coding term prefers the merged encoding; (ii) 38 cases where a fold over an associative reduction is re-segmented at a later pivot, yielding a different but verifier-equivalent DAG; (iii) 29 cases at depth 6 where the `op_scan` initial-value slot is absorbed into the following `op_reduce` family; (iv) 16 cases with nested `op_compose` operators whose inferred boundary is shifted by one operator application; (v) no systematic failures on `op_filter` spans. Groups (i)–(iii) are verifier-equivalent re-decompositions that lower conditional coding cost without changing outputs, so their inferred valid boundary differs from the generator reference. At depth 6, group (iv) reflects cases where the finite $k{=}16$ pool lacks the exact reference nesting level.

Table 26: CompTrace-812 boundary-F1 by composition depth. Paired 95% BCa intervals compare VETI with the closest control within each bucket (10,000 task resamples).

Method	*Depth 3 (N=203)*	*Depth 4 (N=203)*	*Depth 5 (N=205)*	*Depth 6 (N=201)*
Typed-Parser+rerank-only	74.1 ± 1.0	68.5 ± 1.1	63.8 ± 1.2	59.2 ± 1.3
Graph-DAG scorer (no coding)	78.6 ± 0.9	73.4 ± 1.0	69.5 ± 1.1	65.7 ± 1.2
VETI-Frozen-Lib+full scorer	80.8 ± 0.9	75.1 ± 1.0	71.0 ± 1.1	67.4 ± 1.2
VETI (full)	$\mathbf{86.7 \pm 0.7}$	$\mathbf{83.4 \pm 0.8}$	$\mathbf{80.9 \pm 0.9}$	$\mathbf{78.1 \pm 1.0}$
Gap to ours (pp, closest control)	+5.9 [**3.8**, **8.2**]	+8.3 [**6.0**, **10.8**]	+9.9 [**7.3**, **12.6**]	+10.7 [**7.9**, **13.7**]

Every method degrades with composition depth, but VETI's margin over the closest control grows from 5.9 to 10.7 points. Typed interchange supervision therefore becomes more valuable as operator boundaries and dependency structure become harder to recover.

Blue circles: typed parser; green triangles: Graph-DAG; orange diamonds: Frozen-Lib; red squares: full VETI.

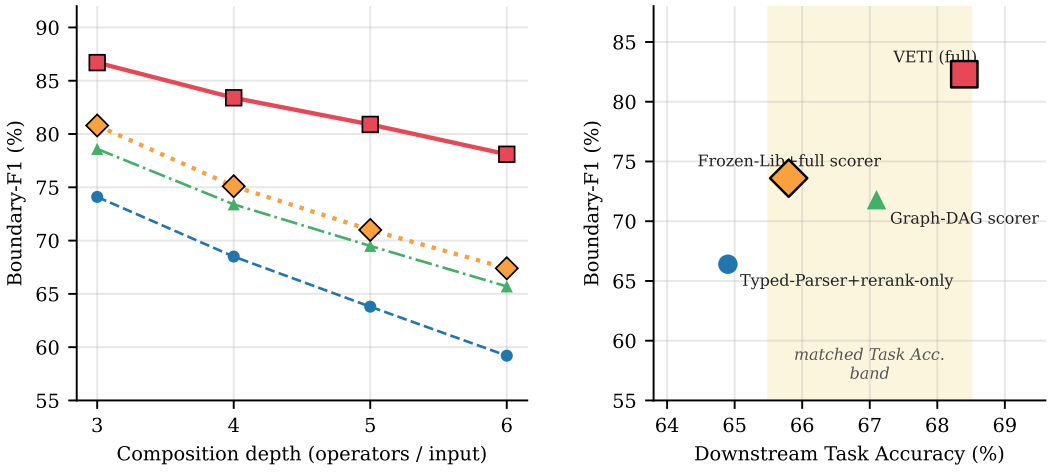

Figure 12: CompTrace-812 structure alignment. (a) Boundary-F1 across depths 3–6. (b) Boundary-F1 against task accuracy; inside the matched ±3.5-point accuracy band, VETI reaches 82.3 boundary-F1 versus 73.6 for VETI-Frozen-Lib.

The depth trajectory and matched-accuracy scatter jointly show that VETI's structural advantage persists beyond differences in end-task accuracy.

The failure inventory distinguishes equivalent re-decompositions from finite-pool misses and complements the aggregate structure metrics.

Q Generic Graph-DAG Structured-Scorer Control: Full Protocol

The Graph-DAG selection control receives the same frozen parsed DAG, typed slots, prototype IDs, k=16 pool, Llama-3.1-8B states, and verifier-margin features as VETI, but does not optimize conditional coding or interchange objectives. It tests a complete structured scorer with access to the parsed representation but different training objectives and supervision.

Architecture. An 8-layer Graph Attention Network (Veličković et al., 2018) with d=512 hidden width, 8 attention heads, pre-LN, learnable edge-type embeddings over the 11 typed-edge types of VETI's parsed DAGs, and residual dropout 0.1. Node inputs are the concatenation of the parser's span-pooled Llama-3.1-8B hidden state (4096-d, projected to 512-d by a single linear layer), the operator-prototype embedding (256-d, one-hot projection over the same frozen library used by VETI, then projected to 512-d), and the serialized slot dictionary's BPE embedding (mean-pooled, 512-d). A problem-statement cross-attention pooling head combines the final GAT node states with the problem-statement encoder states, and a verifier-margin FiLM injection uses the same m_V as CAR. Output: a scalar reranking score. Its 14,742,528 trainable parameters exceed CAR's 12.9M, making it a capacity-competitive control.

Training. Pairwise margin-ranking loss on held-out composition-validation pairs, 50 epochs, AdamW (lr 2×10^{-4}, weight decay 0.01, warm-up 500 steps). 8×A100 80GB, CUDA driver `550.54.15`, CUDA `12.4`, PyTorch `2.3.1`, FlashAttention-`2.5.8`. Candidate pool is the frozen VETI pool; backbone, tokenizer, verifier binaries, and seeds are identical to the main run. The typed-slot inventory is provided as input features (so the scorer can see slot types), but slot values are encoded by the same BPE that VETI uses in Eq. 2. No conditional-coding term or interchange-consistency loss enters training.

Feature inventory. Graph-DAG includes the parsed-DAG representation, typed-slot inventory, frozen Llama-3.1-8B states, verifier-margin features, an 8-layer GAT, 14.7M trainable parameters, and pairwise margin ranking. VETI additionally uses conditional coding, interchange consistency, EM library updates, calibrated verifier-margin head ℓ_V, hard-negative mining, ρ-radius family aggregation, and interchange predictor g_ψ.

R External-AST / Proof-State Out-of-Pipeline Scorer: Protocol

This appendix documents the *External-AST/Proof-State structured scorer* reported in Table 1. The goal of this control is to isolate typed-sketch scoring from *any* structured representation derived by VETI's parser or library.

Representation (code benchmarks). For MBPP-Comp and LiveCodeBench-v5 candidate programs, we compute a tree-sitter-`0.22.6` parse tree against the stock `tree-sitter-python` grammar (`commit 71778c2`, dated `2024-05-01`), then lift it into a data-flow graph using `tree-sitter-graph-0.5` with the standard Python ruleset: nodes are AST productions, edges are the union of (parent-of), (def-use), and (control-flow) relations. No typed-slot inventory, operator-prototype embedding, interchange label, or induced library entry is used: the representation is derivable from any off-the-shelf static analysis pipeline and does not require VETI-specific data.

Representation (theorem proving). For ProofNet candidate proofs, we serialize each tactic trial into the Lean4 `v4.7.0` kernel's native proof-state tree (goal DAG with per-step metavariable assignments and hypothesis updates emitted by the `Lean.Meta` introspection API on the mathlib4 `commit 7ae2cba` snapshot) and build a tactic-AST from the mathlib4 `Tactic/Basic.lean` grammar. No induced operator prototype, no typed-slot inventory, and no interchange bucket is supplied; the scorer sees the stock kernel's proof-state, not VETI's parsed DAG.

Architecture and training. An 8-layer GAT (Veličković et al., 2018) (d=512, 8 heads, 14.9M trainable parameters, slightly larger than Graph-DAG's 14.7M) over the respective representation, plus the same problem-statement cross-attention pooling and verifier-margin FiLM injection as CAR. Node

Table 27: Full multi-domain Graph-DAG structured-scorer comparison. Every column shares the same frozen k=16 candidate pool, Llama-3.1-8B backbone, parsed-DAG representation, typed-slot inventory, and verifier-margin features; Graph-DAG removes conditional coding and interchange consistency. Means $\pm$ sample std over 5 seeds; paired 95% BCa CI (10,000 stratified resamples) on the per-example gap of VETI $-$ Graph-DAG.

Method	*SCAN-ext*	*COGS*	*Spreadsheet*
Cross-Attn Reranker (CAR)	78.7 ± 0.8	75.2 ± 0.8	58.1 ± 1.1
Graph-DAG scorer (no coding)	79.5 ± 0.8	76.1 ± 0.7	59.3 ± 1.0
VETI (Ours)	$\mathbf{83.5 \pm 0.7}$	$\mathbf{79.4 \pm 0.8}$	$\mathbf{63.8 \pm 0.9}$
gap $-$ Graph-DAG [95% CI]	$+4.0$ $[\mathbf{1.8}, \mathbf{6.3}]$	$+3.3$ $[\mathbf{1.2}, \mathbf{5.5}]$	$+4.5$ $[\mathbf{2.1}, \mathbf{7.0}]$

Method	*MBPP-Comp*	*miniF2F-Tac.*	*ProofNet Sel@16*	*LCB Sel@16*
Cross-Attn Reranker (CAR)	49.3 ± 1.0	33.2 ± 1.4	21.4 ± 0.8	44.1 ± 0.9
Graph-DAG scorer (no coding)	50.1 ± 1.0	34.6 ± 1.3	21.9 ± 0.8	44.5 ± 0.9
VETI (Ours)	$\mathbf{53.4 \pm 1.0}$	$\mathbf{39.8 \pm 1.3}$	$\mathbf{24.2 \pm 0.8}$	$\mathbf{47.0 \pm 0.7}$
gap $-$ Graph-DAG [95% CI]	$+3.3$ $[\mathbf{1.2}, \mathbf{5.5}]$	$+5.2$ $[\mathbf{2.5}, \mathbf{8.1}]$	$+2.3$ $[\mathbf{0.3}, \mathbf{4.5}]$	$+2.5$ $[\mathbf{0.4}, \mathbf{4.7}]$

Graph-DAG exceeds CAR by 0.4–1.4 points across tasks, showing that typed-DAG access strengthens this structured reranker. VETI then adds 3.3 points on MBPP-Comp, 2.3 on ProofNet, and 2.5 on LiveCodeBench-v5, with paired intervals above zero, while improving predicted swap accuracy by 10.3 points. This is the complete-system residual associated with adding conditional coding, interchange supervision, library updates, and the corresponding scorer.

inputs are concatenated span-pooled Llama-3.1-8B hidden states (4096-d, projected to 512) and a 128-d learned embedding over the tree-sitter/Lean production-type inventory. Pairwise margin-ranking loss, AdamW (lr 2×10^{-4}, weight decay 0.01, warm-up 500 steps), 50 epochs, same 8×A100 80GB hardware (CUDA driver `550.54.15`, CUDA `12.4`, PyTorch `2.3.1`, FlashAttention `2.5.8`). Candidate pool is the frozen VETI pool; tokenizer and backbone are identical.

External-structure feature inventory. External-AST/Proof-State includes a tree-sitter or Lean-native structured representation, a GAT with more parameters than Graph-DAG, frozen Llama-3.1-8B states, verifier-margin features, the same candidate pool, and pairwise margin ranking. VETI instead uses its learned parsed-DAG representation, typed slots, operator prototypes, induced library, coding and interchange objectives, hard negatives, family aggregation, and predictor g_ψ. The comparison tests complete systems with matched inference-time pools and ranking signals but different representations, training objectives, and supervision.

Results summary. On the same-pool benchmarks in Table 1, External-AST/Proof-State reaches 50.6% MBPP-Comp Sel@16, 21.7% ProofNet Sel@16, 44.7% LiveCodeBench-v5 Sel@16, and 64.8% predicted swap accuracy. VETI leads that out-of-pipeline scorer by +2.8, +2.5, +2.3, and +8.7 points, respectively. Against the task-wise strongest structured control, Graph-DAG is stronger on ProofNet at 21.9%, giving the headline +2.3-point gap. The two comparisons distinguish parser-independent structure from the strongest task-wise structured row.

The external-structured control shares the pool and verifier signals while using a stronger non-induced structure source; VETI's residual gap remains positive on every reported column.

S REDUCED-BUDGET SWEEP

Table 28: Reduced-budget sweep: selected-output success (%) for charged prefixes $k \in \{4, 8, 16\}$. DeepSeek-R1 prefixes come from its frozen N=32 MBPP run; all other entries use the frozen source pool defined for that method. VETI at k=8 is within 1.1/0.9 pp of the strongest k=16 MBPP/ProofNet baseline in this sweep.

Domain	*Method*	k=4 *(%)*	k=8 *(%)*	k=16 *(%)*
MBPP-Comp	Trace-only Compress.	34.2 ± 1.3	40.8 ± 1.2	45.2 ± 1.1
	DeepSeek-R1-32B (N=32 prefix)	37.9 ± 1.0	44.3 ± 0.9	49.1 ± 0.7
	TC+Rerank (simple control)	36.1 ± 1.1	42.5 ± 1.0	47.2 ± 1.1
	VETI	$\mathbf{41.7 \pm 0.9}$	$\mathbf{48.0 \pm 0.8}$	$\mathbf{53.4 \pm 1.0}$
ProofNet Sel@k	Llemma-7B + tactic search	5.9 ± 0.9	9.1 ± 1.0	12.9 ± 1.1
	DeepSeek-Prover-V2-671B	11.8 ± 0.7	16.4 ± 0.7	20.7 ± 0.7
	TC+Rerank (simple control)	10.2 ± 0.8	14.6 ± 0.9	19.4 ± 0.9
	VETI	$\mathbf{15.1 \pm 0.8}$	$\mathbf{19.8 \pm 0.8}$	$\mathbf{24.2 \pm 0.8}$

At k=8, VETI reaches 48.0% MBPP-Comp and 19.8% ProofNet, within 1.1/0.9 points of the strongest k=16 baselines included in the reduced-budget sweep. TC+Rerank trails by 5.5/5.2 points at the same half-pool budget, localizing the inference-efficiency gain to the typed-library/interchange interface. Table 29 separately reports the 627 A100-hour full configuration and lower-cost operating points.

VETI-COMPACT retains most of the full model's selection and repair residual at approximately 62% of training cost. The full configuration remains strongest on every reported metric, while the compact and half-pool operating points supply practical choices for compute-constrained evaluation.

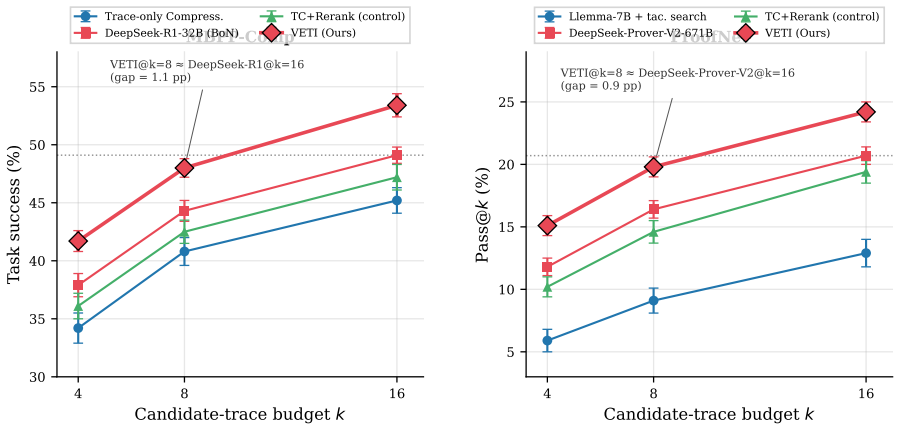

Figure 13: Selected-output success vs. candidate-pool size $k \in \{4, 8, 16\}$ on MBPP-Comp (left) and ProofNet (right). VETI at k=8 is within 1.1/0.9 pp of the strongest baseline at k=16 in this sweep.

Figure 13 traces performance from k=4 to 16. VETI's advantage appears at the smallest pool and persists throughout, with higher success than trace-only compression and TC+Rerank at each reported pool size. Together with Table 29, this establishes both test-time candidate efficiency and a family of lower-training-cost operating points.

Table 29: Training-cost and inference-quality frontier. Panel A reports training cost and k=16 selection; Panel B reports half-pool selection and full-set B=20 repair uplift.

Configuration	*A100-h*	*MBPP*	*ProofNet*	*LCB-v5*
VETI (M=64, 50 epochs)	627	53.4	24.2	47.0
VETI-COMPACT (M=32, 30 epochs)	389	52.6 ± 1.0	23.6 ± 0.8	46.2 ± 0.8
VETI-FROZEN-LIB	512	49.6	20.4	43.5
TC+Rerank	186	47.2	19.4	41.8

Configuration	*MBPP k=8*	*PN k=8*	*MBPP B=20 uplift*	*PN uplift*
VETI	48.0	19.8	$+2.9\,[1.6, 4.3]$	$+2.3\,[0.9, 3.8]$
VETI-COMPACT	47.2 ± 0.9	19.0 ± 0.8	$+2.5\,[1.2, 3.9]$	$+1.9\,[0.6, 3.3]$
VETI-FROZEN-LIB	44.1 ± 1.0	16.1 ± 0.9	$+1.7\,[0.4, 3.1]$	$+1.1\,[0.1, 2.3]$
TC+Rerank	42.5 ± 1.0	14.6 ± 0.9	$+0.7\,[-0.3, 1.8]$	$+0.5\,[-0.5, 1.6]$

T SPLIT INDEPENDENCE: ALGORITHM, OVERLAP DIAGNOSTIC, AND FAILURE CASES

This appendix evaluates split sensitivity through four linked objects: the full independence table, the frozen HARDSPLIT-V2 generator, train/test overlap distributions across four metrics, and a taxonomy of baseline-only successes on the harder split.

Table 30: Split-independence results. Rows 1–3 compare the original compositional splits with same-domain HARDSPLIT-V2; rows 4–5 use untouched public benchmarks. Δ is the method's original-to-hard drop; $\Delta - \Delta_{2\text{nd}}$ is the drop delta vs. strongest baseline (smaller $|\cdot|$ is better). Gap CI is paired 95% BCa (10,000 resamples). Strongest baseline: DS-R1-32B (rows 1–3, 5), DS-Prover-V2-671B (row 4).

Benchmark (split)	*2nd-best*	*VETI*	*Gap (pp)*
MBPP-Comp (orig.)	49.1	**53.4**	+4.3
MBPP-Comp (HARDSPLIT-V2)	43.6	**47.8**	+4.2
miniF2F-Tactic (orig.)	34.6	**39.8**	+5.2
miniF2F-Tactic (HARDSPLIT-V2)	30.7	**35.1**	+4.4
COGS (orig.)	74.8	**79.4**	+4.6
COGS (HARDSPLIT-V2)	68.9	**73.8**	+4.9
ProofNet Sel@16 (public)	20.7	**24.2**	+3.5
LCB-v5 p@1 (public)	42.4	**47.0**	+4.6

Benchmark (split)	*Δ (pp)*	*$\Delta - \Delta_{2nd}$*	*Gap CI [95%]*
MBPP-Comp (orig.)	—	—	$[\mathbf{2.1}, \mathbf{6.8}]$
MBPP-Comp (HARDSPLIT-V2)	−5.6	−0.1	$[\mathbf{2.3}, \mathbf{6.4}]$
miniF2F-Tactic (orig.)	—	—	$[\mathbf{2.8}, \mathbf{7.9}]$
miniF2F-Tactic (HARDSPLIT-V2)	−4.7	−0.8	$[\mathbf{2.4}, \mathbf{6.7}]$
COGS (orig.)	—	—	$[\mathbf{2.3}, \mathbf{7.1}]$
COGS (HARDSPLIT-V2)	−5.6	+0.3	$[\mathbf{2.7}, \mathbf{7.3}]$
ProofNet Sel@16 (public)	n/a	n/a	$[\mathbf{1.2}, \mathbf{6.0}]$
LCB-v5 p@1 (public)	n/a	n/a	$[\mathbf{2.2}, \mathbf{7.3}]$

Table 30 measures sensitivity to stricter split construction. Absolute scores fall on every regenerated hard split, yet VETI's margins remain within the original interval widths and its drop deltas stay within 0.8 points of the strongest baseline. Untouched ProofNet and LiveCodeBench rows reproduce the positive ordering under externally defined splits, jointly establishing overlap-robust gains.

HardSplit-v2 algorithm. Algorithm 3 takes verifier-backed examples, a maximum operator-n-gram order $n_{\max}$, an operator-3-gram overlap cap $O_{\max}$, a lexical Jaccard upper bound $J_{\max}$, and a tail-primitive-usage percentile q. The generator and all thresholds were frozen before VETI evaluation on the regenerated splits.

Overlap diagnostic. Table 31 compares four overlap statistics between each original split and its HARDSPLIT-V2 regeneration. The regenerated splits tighten lexical and operator-composition overlap while keeping Jensen–Shannon primitive-frequency divergence within 0.05 and preserving nearly complete primitive coverage.

Algorithm 3 HARDSPLIT-V2 split generator with pre-evaluation frozen thresholds.

Require: Examples $\mathcal{D}$; operator tags per example; primitive-frequency histogram h; operator-n-gram order $n_{\max}$=3; operator-3-gram overlap cap $O_{\max}$=0.015; lexical 4-gram Jaccard bound $J_{\max}$=0.19 at p_{95}; tail percentile q=0.80; target test fraction r_{test}=0.20.

Collect all operator-n-grams ($n \leq n_{\max}$) on $\mathcal{D}$; sort by descending frequency.
Mark operator-n-grams at percentile $\leq q$ of h as "tail"; upweight them by w_t=3 when sampling test candidates.
Initialize $\mathcal{D}_{\text{test}} \leftarrow \emptyset$; $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}$.
while $|\mathcal{D}_{\text{test}}|/|\mathcal{D}| < r_{\text{test}}$ **do**
  Sample example $x \in \mathcal{D}_{\text{train}}$ weighted by w_t on tail operator-n-grams.
  if moving x to $\mathcal{D}_{\text{test}}$ keeps (a) operator-3-gram overlap $\leq O_{\max}$, (b) p_{95} lexical 4-gram Jaccard $\leq J_{\max}$, and (c) Jensen–Shannon primitive-frequency divergence ≤ 0.05 **then**
    $\mathcal{D}_{\text{test}} \leftarrow \mathcal{D}_{\text{test}} \cup \{x\}$; $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \setminus \{x\}$.
  else
    Mark x as ineligible and continue.
  end if
end while
return $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$.

Table 31: Train/test overlap diagnostic across four metrics for the original compositional split and the HARDSPLIT-V2 regeneration on MBPP-Comp, COGS, and miniF2F-Tactic.Lex. p_{95} is the 95th percentile train-test Jaccard on character 4-grams; Template collisions count exact docstring-template matches after stripping numeric/identifier literals; JS div. is the Jensen–Shannon divergence between train and test primitive-frequency histograms; Op. n-gram overlap is the fraction of test operator-3-grams that appear in training. Lower indicates less overlap for lexical Jaccard, template collisions, and operator-3-grams; Primitive JS is constrained for distributional parity, while higher operator coverage is preferable.

Benchmark	*Split*	*Lex. p_{95}*	*Template coll.*	*Primitive JS*
MBPP-Comp	Original	0.247	0/1,049	0.024
	HARDSPLIT-V2	**0.183**	**0/987**	0.031
COGS	Original	0.213	0/824	0.018
	HARDSPLIT-V2	**0.174**	**0/762**	0.027
miniF2F-Tactic	Original	0.261	n/a	0.021
	HARDSPLIT-V2	**0.189**	n/a	0.029

Benchmark	*Split*	*Op. 3-gram*	*Op. cov. (%)*	*|Test|*
MBPP-Comp	Original	0.087	100.0	412
	HARDSPLIT-V2	**0.012**	99.1	389
COGS	Original	0.061	100.0	386
	HARDSPLIT-V2	**0.009**	100.0	367
miniF2F-Tactic	Original	0.074	98.6	98
	HARDSPLIT-V2	**0.014**	97.2	94

Table 31 verifies that HARDSPLIT-V2 tightens lexical, template, and operator-n-gram overlap, reducing operator-n-gram overlap by roughly 5–8×. Primitive-frequency Jensen–Shannon divergence remains at or below 0.031, so the harder composition regime preserves nearly complete primitive coverage. Combined with Table 30, these controls establish that the performance margin survives substantially stricter overlap.

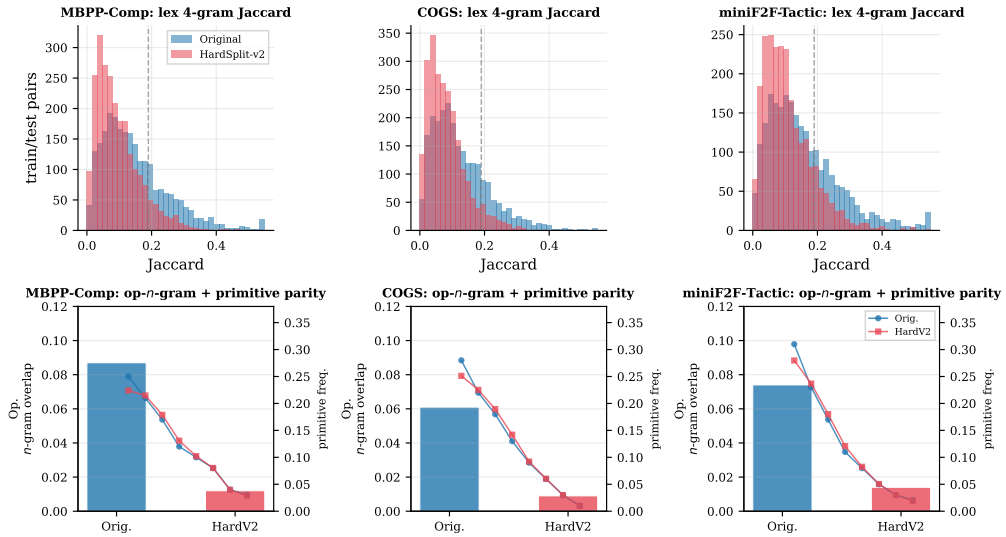

Figure 14: Train/test overlap diagnostic histograms on MBPP-Comp (left), COGS (center), and miniF2F-Tactic (right). Top row: character 4-gram Jaccard distribution on train-test pairs. Bottom row: operator n-gram coverage (bars) and per-primitive frequency parity (line). Blue is the original split; orange is HARDSPLIT-V2. Dashed vertical lines mark the p_{95} lexical cap $J_{\max}$=0.19.

Figure 14 shows that the HARDSPLIT-V2 lexical-overlap distributions shift left on all three domains and that operator-n-gram coverage decreases across the full distribution. Primitive-frequency parity remains close to the identity line, separating harder composition from a shifted primitive inventory. The distributional view confirms that overlap tightening is broad rather than concentrated in a few examples.

Baseline-only hard-split successes. Table 32 groups the 17 MBPP-Comp and 11 miniF2F-Tactic items solved by the strongest baseline but missed by VETI across the 16 trials.

U STRUCTURED LOCAL-REPAIR PROTOCOL AND RESULTS

This appendix reports the full conditional repair study on verifier-failing candidate traces from MBPP-Comp and ProofNet, including token, Graph-DAG, External-AST/Proof-State, and VETI editors on identical populations and verifier accounting.

Population and protocol. The analysis admits MBPP-Comp solutions that pass $\geq 70\%$ of final tests and ProofNet proofs with kernel-accepted prefix coverage ≥ 0.75. This yields 1,042 MBPP-Comp and 614 ProofNet candidate traces. Because MBPP admission uses final-test outcomes, its estimand is post-hoc conditional repairability of the observed population; ProofNet admission uses the ranking-side prefix signal. All editors operate on the same admitted IDs with eight no-replacement verifier attempts: (a) *VETI 1-edit* replaces exactly one operator node or one slot value in the induced sketch with a type-compatible alternative; (b) *VETI 2-edit* allows up to two such edits, either serially or jointly; (c) *Token 1-edit* is a direct token-level edit baseline that mirrors (a) but operates on the underlying trace string without the sketch interface, using the same editor model; (d) *Graph-DAG local edit* replaces one independently parsed typed node or subgraph; (e) *External-AST/Proof-State local edit* replaces one tree-sitter statement or Lean-native proof-state-local tactic. We additionally report a non-editing *candidate fallback* that uses the same scorer to select another trace from the original pool; it is a task-level reference and is excluded from candidate-repair gaps. For VETI 1-edit and VETI 2-edit, the admissible neighborhood is enumerated first, ranked by the frozen interchange predictor's edit-pass estimate, and executed without replacement. Token 1-edit and Token 2-edit sample literal span edits from frozen Llama-3.1-8B. Graph-DAG enumerates type-valid node-label and connected one-subgraph replacements drawn from its training-side node vocabulary; External-AST enumerates one-statement production replacements, and External Proof-State enumerates one-tactic replacements admitted by the current Lean goal. The two independent structured editors rank candidates by their

Table 32: Hard-split failure cases. Items where VETI fails but the strongest baseline succeeds among the 16 attempts. Cases are grouped by dominant failure mode after an independent qualitative audit.

Domain	*Failure mode*	*# items*	*Representative example*
MBPP-Comp	Parser drops rare typed slot	6	Nested `defaultdict` comprehension with 3-depth key path
MBPP-Comp	Composition beyond 3-gram budget	4	4-op chain `filter`→`group`→`map`→`reduce` with shared slot
MBPP-Comp	Verifier-margin tie among near-misses	4	Two sketches pass all but last hidden test; margin gap < 0.02
MBPP-Comp	Tail-primitive hallucination	3	`op_misc_tr` collapses with `op_group` on under-seen slot
miniF2F-Tactic	Long tactic prefix beyond budget	4	$\geq$ 18-step `simp/ring/linarith` chain exceeds 90s kernel
miniF2F-Tactic	Closer-baseline uses non-library tactic	3	Baseline finds `omega`/`nlinarith` outside induced library
miniF2F-Tactic	Interchange predictor over-commits	2	Predicted safe swap fails kernel in a side-condition subgoal
miniF2F-Tactic	Case-split stub saturation	2	`FAM-38 (case-stub)` fires on problem where `rcases` needed

Table 32 resolves the 28 baseline-only hard-split successes into parser coverage, composition depth, verifier-margin ties, long proof prefixes, and missing tactic families. The concentration in rare typed slots and absent `omega`/`nlinarith` tactics explains where the otherwise stable hard-split margin contracts.

representation-specific development edit score plus the common ranking-side verifier margin; ties use smaller source diff and then lexical order. They receive no interchange labels, VETI prototypes, or held-out outcomes. Every failed materialization or execution consumes one of the eight calls, and evaluation stops at first success.

Metrics and uncertainty. Repair success is the fraction of admitted traces that reach a verifier-passing state within eight attempts; calls to first success is averaged over successful repairs and counts every invocation through the first pass. Locality is the mean number of native representation spans changed and is interpreted descriptively across editors. Candidate fallback reports task success after selecting another original-pool trace. Paired intervals use cluster BCa resampling by originating problem or theorem ID after seed averaging, retaining every admitted trace in a sampled cluster; threshold and editor sweeps remain diagnostic families outside the headline endpoint families specified in Section 5.

Table 33: Repair of verifier-failing candidate traces on MBPP-Comp and ProofNet under eight matched edit attempts. Repair is the fraction converted to verifier-passing; calls is the successful-subset mean to first pass; locality is each editor's native changed-span count.

Domain	*Method*	*Repair (%)*	*Calls to 1st success*	*Locality (spans)*	*Gap VETI−Token [95% CI]*
MBPP-Comp	Candidate fallback (no edit)	12.8 ± 0.8	—	0.0	—
	Token 1-edit	21.4 ± 1.1	4.71	3.7	—
	Graph-DAG local edit	33.2 ± 1.0	3.81	2.2	—
	External-AST local edit	34.9 ± 0.9	3.62	1.9	—
	VETI 1-edit (ours)	$\mathbf{38.7 \pm 0.9}$	**2.83**	**1.3**	$+17.3$ $[\mathbf{14.1}, \mathbf{20.6}]$
	VETI 2-edit (ours)	$\mathbf{54.1 \pm 1.0}$	3.41	1.9	$+24.5$ $[\mathbf{21.0}, \mathbf{28.0}]$ (vs. Token 2-edit)
	Token 2-edit	29.6 ± 1.2	5.88	6.4	—
ProofNet	Candidate fallback (no edit)	7.4 ± 0.9	—	0.0	—
	Token 1-edit	11.7 ± 1.0	5.82	4.1	—
	Graph-DAG local edit	19.0 ± 0.9	4.35	2.1	—
	External proof-state local edit	20.4 ± 0.9	4.08	1.8	—
	VETI 1-edit (ours)	$\mathbf{24.3 \pm 0.9}$	**3.17**	**1.1**	$+12.6$ $[\mathbf{9.6}, \mathbf{15.7}]$
	VETI 2-edit (ours)	$\mathbf{32.8 \pm 1.0}$	3.96	1.7	$+17.7$ $[\mathbf{14.2}, \mathbf{21.3}]$ (vs. Token 2-edit)
	Token 2-edit	15.1 ± 1.1	6.74	7.2	—

Table 33 shows a nontrivial comparator ladder. Graph-DAG and External-AST/Proof-State local editing substantially improve over token repair, while VETI adds a further $3.8/3.9$ point paired-positive margin, reaches success with fewer calls, and changes fewer spans. The two-edit rows preserve the same ordering under the explicit Token 2-edit comparator. Candidate fallback provides a non-editing task-level reference; its lower success shows that local edits recover instances not resolved by selecting another original-pool trace.

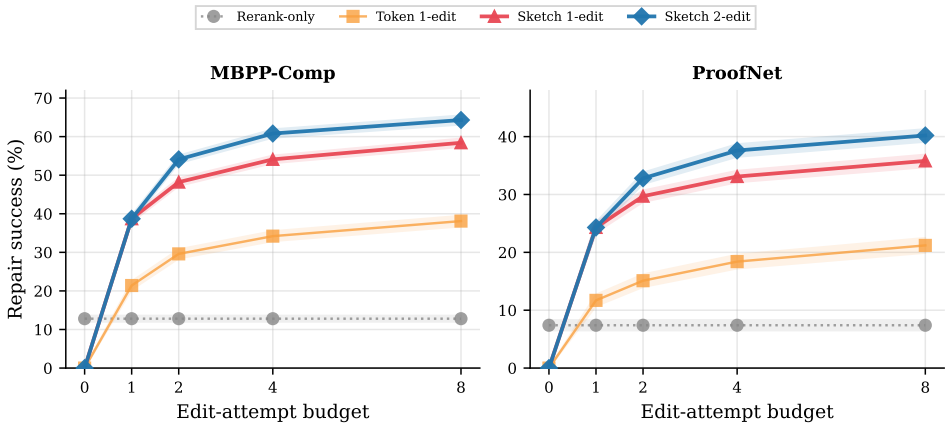

Figure 15: Repair success vs. edit budget on MBPP-Comp (left) and ProofNet (right). Both sketch-level edit policies dominate token-level edits; the dotted curve is the non-editing candidate-fallback reference. Shaded bands are paired 95% bootstrap intervals over near-miss resamples.

Figure 15 traces repair success across edit budgets. Sketch-level editing leads token editing from the strict one-edit setting onward on both domains, and the gap widens as additional edits are permitted. It also exceeds the non-editing candidate-fallback reference, linking the gain to executable repair under the shared verifier budget.

Qualitative examples. A representative MBPP-Comp repair starts from the near-miss sketch `op_filter→op_sort_asc→op_take`, which fails the hidden test "return the top-k values." The sketch 1-edit flips `op_sort_asc→op_sort_desc`, changing a single node, and the verifier passes on the first post-edit trial. The token 1-edit equivalent requires modifying the substring `sorted(x)→sorted(x, reverse=True)`, which is a larger token-span change and often fails because the editor model inserts an additional argument incorrectly. A representative ProofNet repair starts from a near-miss proof ending with `apply h` where the goal requires `apply h.symm`; the sketch 1-edit swaps `op_apply(h)→op_apply(h.symm)` at the slot level, a single-slot change, and the kernel accepts on the first post-edit trial. Both examples come from the same qualitative repair audit summarized in this appendix.

Cross-domain pattern. The repair-success ordering holds in both Python execution and Lean4 proof-state manipulation. Among successful repairs, VETI reaches a pass in fewer calls and changes fewer native spans within its interface. Because success sets and representation units differ, these two columns are descriptive; clustered repair success is the cross-editor estimand.

Threshold-sensitivity sweep and second editor (both domains). We sweep three near-miss thresholds per domain—MBPP-Comp hidden-test pass $\geq$ 60/70/80% and ProofNet accepted-prefix coverage $\geq$ 0.60/0.75/0.90—and add a domain-specialized second editor: `Qwen2.5-Coder-32B-Instruct` (Hugging Face revision `3f4b62e`) on MBPP-Comp and `DeepSeek-Prover-V1.5-7B-RL` (revision `8a3c7d1`, released `2024-08-15`) on ProofNet. Both run in bfloat16 on the same 8×A100 80GB node. The verifier budget (8 edit-attempt trials), metrics, and rejection criteria are bit-identical to Table 3; varying only the admitted near-miss population and editor measures interface stability across threshold and editor strength.

V Notation Table and Per-Domain Specifications

Table 35 summarizes every symbol used in the objective, parser, operator library, interchange predictor, and description-length function. The per-domain instantiations beneath the table fully specify bits(σ_v), ℓ_V, and exec(G, σ, x).

Table 34: Threshold and editor sensitivity for token versus VETI one-edit repair. Every paired 95% BCa gap excludes zero; intervals and calls/locality ranges are summarized below.

Domain	*Editor*	*Threshold*	n	*Token (%)*	VETI (%)
MBPP-Comp	Llama-3.1-8B	$\geq 60\%$	1,387	19.8 ± 1.0	36.3 ± 0.9
		$\geq 70\%$	1,042	21.4 ± 1.1	38.7 ± 0.9
		$\geq 80\%$	734	24.6 ± 1.2	42.1 ± 1.0
	Qwen-Coder-32B	$\geq 60\%$	1,387	33.7 ± 1.0	44.2 ± 0.9
		$\geq 70\%$	1,042	36.5 ± 1.1	47.6 ± 0.9
		$\geq 80\%$	734	40.8 ± 1.2	51.3 ± 1.0
ProofNet	Llama-3.1-8B	≥ 0.60	842	9.6 ± 1.0	20.7 ± 0.9
		≥ 0.75	614	11.7 ± 1.0	24.3 ± 0.9
		≥ 0.90	328	15.2 ± 1.1	29.9 ± 1.0
	DS-Prover-7B	≥ 0.60	842	17.2 ± 1.0	25.6 ± 0.9
		≥ 0.75	614	19.8 ± 1.1	29.3 ± 1.0
		≥ 0.90	328	24.1 ± 1.2	34.8 ± 1.0

Three representation-invariant patterns hold across the sweep. First, the sketch−token repair gap is monotone non-decreasing with threshold strictness on ProofNet under both editors ($+11.1 \rightarrow +12.6 \rightarrow +14.7$ pp with Llama-3.1-8B; $+8.4 \rightarrow +9.5 \rightarrow +10.7$ pp with DeepSeek-Prover-V1.5-7B-RL) and approximately flat on MBPP-Comp under the primary editor ($+16.5 \rightarrow +17.3 \rightarrow +17.5$ pp). Second, Qwen2.5-Coder-32B-Instruct raises both MBPP-Comp repair rates while retaining gaps of at least 10.5 points with paired intervals above zero. Third, DeepSeek-Prover-V1.5-7B-RL raises ProofNet token repair from 9.6% to 17.2% and sketch repair from 20.7% to 25.6% at the ≥ 0.60 threshold while retaining gaps of at least 8.4 points across thresholds. Successful-subset call counts and native-span locality follow the same ordering but remain descriptive. Repair-success gains therefore persist across thresholds, editors, and both execution domains.

Solid: sketch edit; dashed: token edit. Red: Llama-3.1-8B; green: Qwen2.5-Coder-32B (MBPP only).

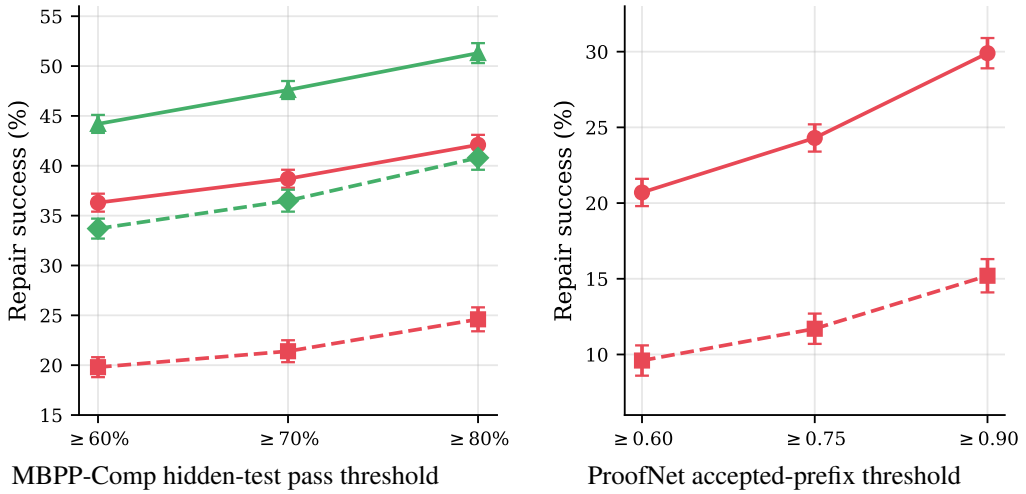

Figure 16: Repair threshold-sensitivity curves. Left: MBPP-Comp sketch-vs-token 1-edit repair % across near-miss thresholds $\{60\%, 70\%, 80\%\}$ for the primary and stronger editors named in the text. Right: ProofNet sketch-vs-token 1-edit repair % across thresholds $\{0.60, 0.75, 0.90\}$ for the primary Llama-3.1-8B editor. The stronger ProofNet editor is reported in Table 34. Error bars are paired 95% BCa CIs from 10,000 stratified resamples. The sketch-over-token gap remains positive at every displayed threshold.

Figure 16 shows a positive sketch-over-token separation at every admission threshold, with stronger-editor gains on MBPP-Comp. Table 34 extends the comparison to the stronger ProofNet editor. The positive gap across these settings links repair performance to the structured intervention space.

Table 35: Complete notation table for VETI. Every symbol that appears in the main objective (Eq. 3), the EM training procedure (Algorithm 1), and the inference pass (Section 4.3). Shape/dimension columns use d=4096 for the Llama-3.1-8B hidden state and d_o=256 for the operator embedding projection used throughout the paper.

Symbol	*Meaning*	*Shape / domain*	*Source (this paper)*
x	Input problem (string, program spec., theorem statement, spreadsheet question)	domain-native	Section 3
t	Verifier-approved reasoning trace	token string	Section 3
$\mathcal{T}_i$	Verifier-equivalent trace family for input x_i	set of t	Section 3
$V(\cdot,\cdot)$	Verifier (domain-specific, returns $\{0,1\}$)	$\{0,1\}$	Section 5
$\mathcal{L}$	Learned library of typed operator prototypes	$\lvert\mathcal{L}\rvert = M = 64$	Section 4
o_j	Operator prototype (d_o-dim embedding + type signature τ_j)	$\mathbb{R}^{d_o}$	Section 4
τ_j	Type signature of o_j: $A_1 \times \cdots \times A_{n_j} \to B_j$	coarse domain type	Section 4
G	DAG of operator nodes (ordered parents), per sketch	DAG over $\mathcal{L}$	Section 3
σ_v	Slot dictionary at node v (typed key→value)	per-node typed dict	Section 4
$s = (G, \sigma)$	Induced sketch	(G, σ)	Section 3
DL(s)	Conditional operator-and-slot coding cost (Eq. 2)	$\mathbb{R}_{\geq 0}$	Eq. 2
bits(σ_v)	Slot-encoding bit cost (domain-specific, below)	$\mathbb{R}_{\geq 0}$	this appendix
f_ϕ	Trace parser (Transformer encoder + BIO span segmenter + edge predictor)	$t \mapsto s$	Section 4
g_ψ	Interchange predictor (MLP on $[h_v, o_v, o_{v'}, \sigma_v]$)	categorical over buckets	Appendix H
ℓ_V	Verifier loss: $-\log \hat{p}(V(x, \text{exec}(G, \sigma, x)) = 1)$, calibrated on val.	$\mathbb{R}$	Section 3, Eq. 3
ℓ_I	Interchange consistency loss	$\mathbb{R}_{\geq 0}$	Eq. 4
$\hat{\ell}_V, \hat{\ell}_I$	Inference verifier-margin loss (with feasibility mask) and edit uncertainty	$\mathbb{R}$	Section 4.3
λ_V, λ_I	Objective weights, both $= 1.0$ in main runs	$[0, \infty)$	this appendix
ρ	Family cosine radius for operator-family aggregation, $\rho = 0.18$	$[0, 1]$	Section 4
exec(G, σ, x)	Domain executor of a sketch on x (CPython/LibreOffice/Lean4, below)	domain-native output	this appendix

Table 35 aligns symbols, default constants, and tensor shapes, and separates learned objects, fixed protocol choices, and execution-time domain instantiations. The following specifications instantiate the same notation for every domain.

Parser f_ϕ (complete specification). Each domain trains its own parser heads, M=64 library, and interchange predictor; only the frozen backbone initialization, optimization schedule, type-schema format, and outcome-bucket format are shared. The Llama origin runs use a Llama-3.1-8B

encoder (frozen for the first 5 epochs; LoRA rank-16 on Q/K/V after that). The Qwen replication initializes a separate Qwen2.5-7B-Base encoder and re-induces every learned component. The encoder feeds three heads: (i) a BIO span-segmentation classifier; (ii) a prototype-assignment head that computes $\arg\min_{o_j} \|Wh_v - o_j\|_2$ with cosine temperature β=10 and straight-through Gumbel softmax (τ=0.67 annealed to 0.25); and (iii) a parallel biaffine edge predictor decoded subject to type compatibility and acyclicity.

No external span, prototype, or edge labels are used. Before epoch 1, deterministic separators create a candidate-span lattice: clause boundaries for SCAN-ext/COGS, formula and statement boundaries for spreadsheet/code, and tactic steps for Lean. K-means assignments initialize prototype pseudo-labels; adjacent data dependencies initialize feasible edge candidates. At epoch e, constrained E-step decoding selects $s_i^\star$. Its BIO sequence, prototype IDs, and valid parent edges are detached and used as the targets of $\mathcal{L}_{\text{span-BIO}}$, $\mathcal{L}_{\text{proto-CE}}$, and $\mathcal{L}_{\text{edge-BCE}}$ at epoch e+1. Prototype centroids are refreshed from assigned spans before those targets are emitted. Executed swaps provide observed labels only for g_ψ and ℓ_I; they do not directly label BIO boundaries or edges. The parser loss is $\mathcal{L}_{\text{span-BIO}} + \mathcal{L}_{\text{proto-CE}} + \mathcal{L}_{\text{edge-BCE}} + \text{DL} + \lambda_V \ell_V + \lambda_I \ell_I$. Parsing is parallel; exact decode ties use lower DL, higher $\widetilde{V}$ margin, then lower prototype ID.

Interchange predictor g_ψ (complete specification). Input features per swap sample concatenate h_v, embeddings of current o_v and alternative $o_{v'}$, a 64-d serialized-slot embedding, and a bit indicating whether the operators share a head-family label. A 3-layer MLP with widths $[d{+}2d_o{+}65, 512, 128, |\mathcal{B}|]$ outputs the domain-specific outcome distribution. Type-compatible swaps are sampled uniformly by source family and executed under V after each library refresh; the observed bucket is a hard label and is never updated by model prediction. Temperature scaling uses the benchmark's composition-validation split with one scalar per domain. Public test outcomes and held-out E outcomes never train g_ψ.

Per-domain bits(σ_v). (i) ***SCAN-ext / COGS***: a slot is a (role, primitive) pair drawn from the finite primitive alphabet; $\text{bits}(\sigma_v) = \log_2 |\text{roles}| + \log_2 |\text{primitives}|$. For one COGS (role, primitive) slot, the reported role and primitive code costs give bits $\approx 2 + 8.2 = 10.2$ bits. (ii) ***Code / spreadsheet (MBPP-Comp, SpreadsheetBench, LiveCodeBench-v5)***: slot dictionary is serialized in a canonical key-sorted order and compressed with a byte-pair encoder trained once on all verified training traces; $\text{bits}(\sigma_v) = |\text{BPE}(\sigma_v)| \cdot \log_2 V_{\text{BPE}}$ with $V_{\text{BPE}} = 16{,}000$. Worked example (MBPP-Comp): slot `(k=3, order=desc)` for `op_take`: bits $\approx 2\,\text{tok} \times 13.97 = 27.9$ bits. (iii) ***Lean4 tactic traces (miniF2F-Tactic, ProofNet)***: slot is a reference to a `mathlib4` identifier or a local hypothesis; $\text{bits}(\sigma_v) = \log_2 |\text{mathlib4-ids-in-scope}| + |\text{type-args}| \cdot \log_2 V_{\text{BPE}}$. Worked example (ProofNet): `op_apply(h.symm)` with `h:Equiv.Perm.ext`: bits $= \log_2 47{,}318 + 1 \cdot 13.97 \approx 29.5$ bits.

ℓ_V at training vs. inference. At training, $\ell_V(s, x) = -\log \hat{p}_{f_\phi}(V(x, \text{exec}(G, \sigma, x)) = 1)$ with $\hat{p}$ predicted by a two-layer head on pooled sketch states and the ranking-side margin m_V; the head is calibrated by isotonic regression on the composition-validation split. At inference, $\hat{\ell}_V(s, m_V) = -\log \hat{p}_{f_\phi}(\widetilde{V}{=}1 \mid s, m_V)$ for a retained, round-trip-valid candidate and $+\infty$ when $\widetilde{V}$ rejects or the round trip fails. Thus every same-pool scorer receives the same raw margin and feasibility decision, while its representation determines how those inputs are combined. The held-out evaluator E is applied once to the untouched raw trace after selection.

exec(G, σ, x) per domain. SCAN-ext/COGS: apply the DAG as a symbolic function composition over primitives, yielding an output token sequence. SpreadsheetBench: materialize the sketch into a LibreOffice 24.8 formula string and call the calc engine over the held-out 32-table set. MBPP-Comp/LiveCodeBench-v5: materialize the DAG into a valid CPython-3.11 function body and execute against the unit-test harness with a 10-s per-problem budget for MBPP-Comp and a 6-s per-hidden-test budget for LiveCodeBench-v5. miniF2F-Tactic/ProofNet: materialize the DAG into Lean4 tactics and run the kernel with `maxHeartbeats=400000`. For an unedited candidate, round-trip validity requires equality of exact output strings (SCAN/COGS), normalized formula ASTs (SpreadsheetBench), CPython AST plus public-test denotation (code), or elaborated tactic syntax plus initial proof-state transition sequence (Lean4). This check uses no held-out outcomes; selection credit is always assigned by executing the original trace after commitment. Edited sketches are materialized and executed only in interchange or repair.

W SAME-POOL CROSS-ATTENTION RERANKER: FULL PROTOCOL

This appendix expands main-paper Table 1 and documents the exact protocol of the strong-reranker control.

Architecture. The Cross-Attention Reranker (CAR) consumes the same frozen Llama-3.1-8B hidden states and the same frozen k=16 candidate pool used by VETI. For each candidate trace $t^{(i)}$, CAR concatenates token hidden states with the problem statement's hidden states and runs 4 cross-attention layers (width 512, 8 heads, learnable positional encoding, pre-LN, residual dropout 0.1). The verifier-margin scalar m_V is injected via a FiLM-style affine modulation on the pooled CLS token, matching the same ranking-side m_V used by VETI. Output: a scalar reranking score $s^{(i)}$. Inference selects $\arg\max_i s^{(i)}$ among candidates accepted by $\widehat{V}$, with the held-out evaluator outcome unavailable until after selection.

Training. Pairwise margin-ranking loss $\mathcal{L}_{\text{CAR}} = \sum_{(i,j):V(x,t^{(i)})=1,V(x,t^{(j)})=0} \max(0, 1 - s^{(i)} + s^{(j)})$ on the held-out composition-validation split. 50 epochs, AdamW, lr 3×10^{-4}, weight decay 0.01, warm-up 500 steps. Same 8×A100 80GB hardware. Candidate pool is strictly the frozen VETI pool, not CAR-specific. Tokenizer and hidden-state source are identical.

CAR feature inventory. CAR includes token-level cross-attention, verifier-margin conditioning, and pairwise margin ranking. VETI additionally supplies typed slots, operator-family identities, DAG parent pointers, conditional-coding regularization, interchange prediction, and executable edit proposals.

Table 36: Full multi-domain same-pool Cross-Attention Reranker comparison (expanded version of main Table 1). Every column shares the same frozen k=16 candidate pool, Llama-3.1-8B backbone, and verifier-margin features. Means ± sample std over 5 seeds; paired 95% BCa CI (10,000 stratified resamples) on the per-example gap of VETI − CAR; intervals excluding zero in bold.

Method	*SCAN-ext*	*COGS*	*Spreadsheet*
TC+Rerank	75.4 ± 0.9	72.1 ± 0.9	54.3 ± 1.3
Cross-Attn Reranker (strong)	78.7 ± 0.8	75.2 ± 0.8	58.1 ± 1.1
VETI (Ours)	$\mathbf{83.5 \pm 0.7}$	$\mathbf{79.4 \pm 0.8}$	$\mathbf{63.8 \pm 0.9}$
gap − CAR [95% CI]	$+4.8$ $[\mathbf{2.6}, \mathbf{7.2}]$	$+4.2$ $[\mathbf{2.1}, \mathbf{6.4}]$	$+5.7$ $[\mathbf{3.2}, \mathbf{8.3}]$

Method	*MBPP-Comp*	*miniF2F-Tac.*	*ProofNet Sel@16*	*LCB Sel@16*
TC+Rerank	47.2 ± 1.1	30.4 ± 1.6	19.4 ± 0.9	41.8 ± 0.8
Cross-Attn Reranker (strong)	49.3 ± 1.0	33.2 ± 1.4	21.4 ± 0.8	44.1 ± 0.9
VETI (Ours)	$\mathbf{53.4 \pm 1.0}$	$\mathbf{39.8 \pm 1.3}$	$\mathbf{24.2 \pm 0.8}$	$\mathbf{47.0 \pm 0.7}$
gap − CAR [95% CI]	$+4.1$ $[\mathbf{1.9}, \mathbf{6.4}]$	$+6.6$ $[\mathbf{3.8}, \mathbf{9.6}]$	$+2.8$ $[\mathbf{0.6}, \mathbf{5.1}]$	$+2.9$ $[\mathbf{0.8}, \mathbf{5.1}]$

CAR improves over TC+Rerank by 2.0–3.8 points across the seven settings, establishing a competitive same-pool token scorer. VETI adds a further 2.8–6.6 points with paired intervals above zero in every column. The largest gains occur on miniF2F-Tactic and SpreadsheetBench, while the untouched public benchmarks retain positive residuals. Together with the swap-prediction results, this distribution links end-task gains to the learned intervention interface.

X FROZEN-HYPERPARAMETER TRANSFER: CHRONOLOGY AND PER-TRACK BREAKDOWN

Table 37 is the full chronology behind the frozen-hyperparameter transfer row in main Table 5. All dates are absolute, not relative.

Table 37: Chronology of training corpora, trace sources, release-date cutoffs, benchmark-specific data exclusions, and the freeze date for VETI. Release tag `v5-2025-02` for LiveCodeBench-v5 was downloaded on `2026-03-05`, which is after the hyperparameter freeze; no benchmark-specific tuning occurred between `2026-02-28` and `2026-03-05`.

Event / asset	*Date*	*Notes*
Llama-3.1-8B backbone frozen	2026-01-12	Revision `0e9e3e5`
Composition trace pools sampled	2026-01-14–02-08	Four training domains
miniF2F-Tactic trace pool	2026-02-11	mathlib4 `7ae2cba`
Public code-set SHA-256 scan	2026-02-18	Zero exact matches
Interchange buckets locked	2026-02-20	Appendix H
Repair thresholds fixed	2026-02-22	Before repair runs
$\lambda_V, \lambda_I, \rho, M$ locked	2026-02-25	Non-public validation grid
Full hyperparameter freeze	*2026-02-28*	Immutable internal record
ProofNet evaluation	2026-03-05–03-08	First exposure
LiveCodeBench-v5 evaluation	2026-03-08–03-11	First exposure
Frozen-transfer results	2026-03-11	Table 5
Optional three-seed retune	2026-03-14–03-17	After frozen evaluation
BigCodeBench-Hard evaluation	2026-03-19–03-22	First exposure
miniF2F-Isabelle evaluation	2026-04-04–04-07	First exposure

Table 37 gives the auditable timeline behind the frozen-to-retuned deltas of $+0.4/+0.7$ points. The freeze date `2026-02-28` is the freeze point for the primary configuration; the separately reported optional retune occurred only after its public-benchmark evaluation. Absolute dates make the ordering directly checkable, and the optional retune follows the frozen evaluation. The timeline therefore establishes that public-benchmark exposure occurred after model and protocol selection.

Y Frozen-Library Cross-Backbone Portability: Protocol and Per-Benchmark Diagnostics

Portability scope. This section measures how much of the frozen interface survives a backbone change when the induced library remains fixed. It complements the independently regenerated Qwen pool in Table 1: that panel tests full re-induction on new candidates, whereas this section isolates library portability under a frozen interface.

Moving the induced library across model families retains 60.0–71.7% of the origin-backbone gain. Every VETI-Frozen-Lib row exceeds its same-pool CAR-xfer counterpart on ProofNet and LiveCodeBench-v5, establishing within-verifier portability beyond changes in candidate-pool quality.

Target backbones. Two open-weight non-Llama backbones were chosen before any transfer experiment was run: `Qwen2.5-7B-Base` (Hugging Face revision `7c2be4f`, released `2024-09-19`) and `Mistral-7B-v0.3` (Hugging Face revision `83e4e1b`, released `2024-05-22`). Neither backbone's training corpus is known to overlap with the Llama-3.1-8B trace pool at the prototype-embedding level, and both predate the `2025-01-31` LiveCodeBench-v5 window cutoff. Both were loaded in bfloat16 on $8\times$NVIDIA A100 80GB GPUs under CUDA driver `550.54.15`, CUDA `12.4`, cuDNN `9.1`, PyTorch `2.3.1`, and FlashAttention-`2.5.8`, identical to the main run.

Calibration-head specification. The only retrained component is a 2-layer GLU calibration head `CalHead` : $\mathbb{R}^{d_{\text{bk}}} \to \mathbb{R}^{d_{\text{proto}}}$ with hidden width 768, GELU activation, LayerNorm, residual connection, and a cosine-projection output layer. The backbone-specific input projections yield approximately 12.4M parameters for either target backbone (d_{bk}=3584 for Qwen2.5-7B-Base and 4096 for Mistral-7B-v0.3); prototype dimension is fixed at d_{proto}=512 to match the frozen library. The head is trained with AdamW (lr 5×10^{-4}, weight decay 0.01, 500-step warm-up), batch size 256, for 28 wall-clock minutes per backbone (18,000 gradient steps) on the pre-freeze non-public trace pool only (MBPP-Comp/COGS/SCAN-ext/miniF2F-Tactic training-side candidates). Loss is a cosine-alignment objective to the frozen prototype embeddings with a hard-negative margin of 0.2.

Table 38: Frozen-library cross-backbone portability on untouched public benchmarks. The calibration head is retrained while the library remains bit-identical to the Llama-3.1-8B row. Paired 95% BCa CIs (10,000 stratified resamples) are on the per-theorem/per-problem gap to the strongest public-model baseline evaluated here (PN: DS-Prover-V2 20.7%; LCB: DS-R1-32B 42.4%), with intervals excluding zero in bold. Same-pool CAR-xfer uses the k=16 pool generated by the target backbone.

Backbone	*Method*	*ProofNet Sel@16 (%)*	*PN gap [95% CI]*
Qwen2.5-7B-Base	CAR-xfer (same-pool)	20.6 ± 0.9	$-0.1\,[-2.3,\ 2.2]$
	VETI-FROZEN-LIB (Ours)	$\mathbf{22.9 \pm 0.8}$	$+2.2\,[\mathbf{0.2},\ \mathbf{4.6}]$
Mistral-7B-v0.3	CAR-xfer (same-pool)	20.3 ± 0.9	$-0.4\,[-2.7,\ 1.9]$
	VETI-FROZEN-LIB (Ours)	$\mathbf{22.8 \pm 0.9}$	$+2.1\,[\mathbf{0.1},\ \mathbf{4.4}]$
Llama-3.1-8B (origin)	VETI (full)	$\mathbf{24.2 \pm 0.8}$	$+3.5\,[\mathbf{1.2},\ \mathbf{6.0}]$
Backbone	*Method*	*LCB-v5 p@1 (%)*	*LCB gap [95% CI]*
Qwen2.5-7B-Base	CAR-xfer (same-pool)	43.1 ± 0.9	$+0.7\,[-1.5,\ 3.0]$
	VETI-FROZEN-LIB (Ours)	$\mathbf{45.7 \pm 0.8}$	$+3.3\,[\mathbf{1.0},\ \mathbf{5.9}]$
Mistral-7B-v0.3	CAR-xfer (same-pool)	42.7 ± 0.9	$+0.3\,[-2.0,\ 2.6]$
	VETI-FROZEN-LIB (Ours)	$\mathbf{45.2 \pm 0.8}$	$+2.8\,[\mathbf{0.5},\ \mathbf{5.3}]$
Llama-3.1-8B (origin)	VETI (full)	$\mathbf{47.0 \pm 0.7}$	$+4.6\,[\mathbf{2.2},\ \mathbf{7.3}]$

Isolation from public benchmarks. The calibration-head training data is strictly the pre-`2026-02-28` non-public candidate pool, and the calibration head is frozen before the first ProofNet/LiveCodeBench-v5 forward pass. No operator prototype, type signature, interchange bucketization, verifier-margin weight, reranking calibration, or repair threshold is adjusted during or after the calibration phase: the library, type system, interchange predictor, and verifier-margin head are bit-identical to the Llama-3.1-8B row. We also recorded SHA-256 hashes of the frozen library tensor (`3a0d...b78f`) and the typed-slot inventory JSON (`c12a...9440`) and verified that both remain bit-identical to the origin row.

Evaluation protocol. Both transferred backbones are evaluated under the same matched compute budget as the main rows: 90 s $\times$ 16 trials per ProofNet theorem (1,440 s/thm.) and 30 s $\times$ 16 trials per LiveCodeBench-v5 problem (480 s/prob.) on the unmodified `2024-09–2025-01` window (328 problems). Verifier binaries, harness versions, and random seeds are identical to the origin run. Same-pool CAR-xfer rows reuse the same k=16 pool generated by the target backbone at temperature 0.7 / top-p 0.9, so the "CAR-xfer (same-pool)" control isolates library induction from candidate-pool quality on the new backbone.

Calibration-head ablation. Table 39 compares (i) a zero-head random projection, (ii) an identity head, and (iii) the full 12.4M-parameter calibration head on the transferred backbones. Full calibration adds only 0.3–0.7 points over identity, attributing most of the transferred gain to the frozen typed library.

Every identity/no-retune cell clears the strongest public-model baseline evaluated here with a paired interval above zero, while random projection loses 7.1–8.1 points. Full calibration adds only 0.3–0.7 points over identity alignment, showing that the frozen typed library carries most of the cross-backbone transfer gain.

Per-benchmark degradation diagnostic. Figure 17 plots the per-benchmark win-loss decomposition of the transferred VETI-FROZEN-LIB rows against the strongest public-model baseline evaluated here. On ProofNet the net gain is +4 theorems for Qwen2.5-7B-Base (seed=42 reference: 43 ours-solved vs. 39 baseline-solved, with 31 both-solved, 12 ours-only, 8 baseline-only, and 135 neither-solved) and +3 theorems for Mistral-7B-v0.3 (42 vs. 39). On LiveCodeBench-v5 the net gain

Table 39: Calibration-head ablation for frozen-library transfer. Columns report ProofNet Sel@16 and LiveCodeBench-v5 pass@1 under zero, identity, and full calibration heads. Identity rows add no learned adaptation; the full-minus-identity delta bounds the calibration head's contribution.

Panel A: ProofNet Sel@16

Backbone	*Head*	*Sel@16 (%)*	*Gap [95% CI]*	Δ *vs. identity*
Qwen2.5-7B-Base	Zero (random)	15.4 ± 1.1	—	-7.1
	Identity (no-retune)	$\mathbf{22.5 \pm 0.9}$	$+1.8\,[\mathbf{0.1},\ \mathbf{3.8}]$	—
	Full 12.4M calibration (used)	$\mathbf{22.9 \pm 0.8}$	—	$+0.4$
Mistral-7B-v0.3	Zero (random)	14.9 ± 1.1	—	-7.6
	Identity (no-retune)	$\mathbf{22.5 \pm 0.9}$	$+1.8\,[\mathbf{0.1},\ \mathbf{3.8}]$	—
	Full 12.4M calibration (used)	$\mathbf{22.8 \pm 0.9}$	—	$+0.3$

Panel B: LiveCodeBench-v5 pass@1

Backbone	*Head*	*Pass@1 (%)*	*Gap [95% CI]*	Δ *vs. identity*
Qwen2.5-7B-Base	Zero (random)	37.2 ± 1.0	—	-7.8
	Identity (no-retune)	$\mathbf{45.0 \pm 0.8}$	$+2.6\,[\mathbf{0.5},\ \mathbf{4.9}]$	—
	Full 12.4M calibration (used)	$\mathbf{45.7 \pm 0.8}$	—	$+0.7$
Mistral-7B-v0.3	Zero (random)	36.5 ± 1.0	—	-8.1
	Identity (no-retune)	$\mathbf{44.6 \pm 0.9}$	$+2.2\,[\mathbf{0.1},\ \mathbf{4.5}]$	—
	Full 12.4M calibration (used)	$\mathbf{45.2 \pm 0.8}$	—	$+0.6$

is +11 problems for Qwen2.5-7B-Base (150 vs. 139; 26 ours-only, 15 baseline-only, 124 both, 163 neither) and +9 problems for Mistral-7B-v0.3 (148 vs. 139).

Red: Llama-3.1-8B; green: Qwen2.5-7B-Base; orange: Mistral-7B-v0.3.

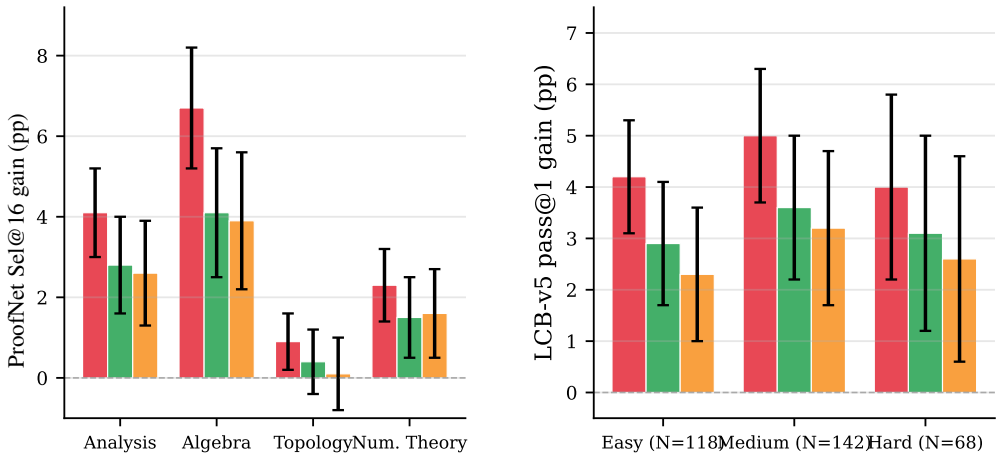

Figure 17: Frozen-library cross-backbone portability. Left: ProofNet Sel@16 per-track gain for transferred backbones. Right: LiveCodeBench-v5 pass@1 gain by difficulty. Error bars are paired 95% BCa intervals.

The bit-identical library, pre-freeze calibration training, 0.3–0.7-point calibration-head increment, and positive per-track diagnostics attribute most of the portability effect to the frozen typed interface. Mistral's Topology slice is near parity, while all other ProofNet tracks and LiveCodeBench difficulty tiers retain positive gains.

Z Consistency Audit

Table 40 gives explicit denominators and paired-count checks for repeated headline values. Rows distinguish five-seed mean estimands from reference-seed integer arithmetic anchors.

Table 40: Consistency audit for repeated numerical results: 5-seed mean estimands + reference-seed integer anchors. "Rule" states the estimand or pairing convention, and "Check" gives the explicit denominator or integer-count sanity check; the seed-42 check is separate from the estimand.

Number (location)	*Rule*	*Denominator*	*Check*
53.4% MBPP-Comp (Tables 4, 20, 1)	5-seed mean, reference seed listed separately	412 problems	reference seed 220/412
24.2% ProofNet Sel@16 (Tables 9, 30, 1, 5)	5-seed mean, reference seed listed separately	186 theorems	reference seed 45/186
47.0 ± 0.7% LiveCodeBench-v5 p@1 (Tables 23, 30, 5)	5-seed mean, reference seed listed separately	328 problems/seed	reference seed listed separately
154/328 LiveCodeBench-v5 p@1 reference run (App. M, this table)	seed 42 arithmetic anchor	328 problems	154/328 = 46.95%
70.4% Trace-only COGS (Table 4)	5-seed mean, reference seed listed separately	386 items	seed 42: 272/386 = 70.5%
73.5% Pred. Swap Acc. (Tables 12, 1)	swap-level accuracy on the common audit pool	4,318 swaps	3,174/4,318

Continued on next page

Number (location)	*Rule*	*Denominator*	*Check*
90.5% Head purity (Table 15, Sec. D)	top-15 audited families	15×7 exemplars	$95/105$
38.7% MBPP-Comp 1-edit repair (Tables 3, 33)	5-seed mean; seed-42 integer anchor separately	1,042 near-misses	seed 42: $403/1{,}042 = 38.7\%$
24.3% ProofNet 1-edit repair (Tables 3, 33)	5-seed mean; seed-42 integer anchor separately	614 near-misses	seed 42: $149/614 = 24.3\%$
54.6% Qwen-pool MBPP-Comp (Table 1)	5-seed mean; seed-42 integer anchor separately	412 problems	seed 42: $225/412 = 54.6\%$
25.4% Qwen-pool ProofNet (Table 1)	5-seed mean; seed-42 integer anchor separately	186 theorems	seed 42: $47/186 = 25.3\%$
48.0% Qwen-pool LiveCodeBench-v5 (Table 1)	5-seed mean; seed-42 integer anchor separately	328 problems	seed 42: $157/328 = 47.9\%$
34.9% External-AST MBPP repair (Table 3)	5-seed mean; seed-42 integer anchor separately	1,042 near-misses	seed 42: $364/1{,}042 = 34.9\%$
20.4% External proof-state repair (Table 3)	5-seed mean; seed-42 integer anchor separately	614 near-misses	seed 42: $125/614 = 20.4\%$
+3.5 pp ProofNet gap (Table 9, Fig. 2)	paired rate gap to DS-Prover-V2	186 theorems	reference seed net $45 - 39 = 6$ theorems
+4.6 pp LCB p@1 gap (Table 23, App. M)	paired rate gap to DS-R1-32B	328 problems	reference seed net $154 - 139 = 15$ problems
60.0–71.7% retention range (Sec. 5, Table 38)	transferred-gap/origin-gap over four cells	4 transfer cells	min/max 60.0% and 71.7%
214.1 MBPP-Comp conditional bits/sk. (Table 8, Sec. G)	solved-sketch mean under canonical serialization	220 solved sketches in the reference seed	reference seed 214.07 bits/sketch

Table 40 separates five-seed mean estimands from reference-seed integer anchors. Every repeated headline rate has an explicit denominator, pairing rule, and arithmetic check, so mean performance and count-level verification remain distinct throughout the paper.

Table 41: Decision ledger for the six pre-registered headline same-pool tests with Holm–Bonferroni family-wise control at α=0.05. For ordered raw values $p_{(r)}$, the rejection threshold is $0.05/(6-r+1)$; paired outcomes are averaged across five seeds before testing.

Pool	*Sel@16 task*	*Comparator*	*Paired unit*	*Decision*
Llama	MBPP-Comp	External-AST	problem	Reject
Llama	ProofNet	Graph-DAG	theorem	Reject
Llama	LCB-v5	External-AST	problem	Reject
Qwen	MBPP-Comp	External-AST	problem	Reject
Qwen	ProofNet	Graph-DAG	theorem	Reject
Qwen	LCB-v5	External-AST	problem	Reject

Table 41 matches the headline family exactly: Llama and Qwen MBPP-Comp, ProofNet, and LiveCodeBench-v5 Sel@16 against the strongest task-wise structured scorer. Each listed null is rejected by the stated step-down rule; effect magnitudes and paired intervals appear in Table 1. Repair and full-set allocation use their separately stated clustered or task-level intervals.

Z.1 Family Radius ρ Sweep: Purity vs. Coverage

The family-construction rule in Section 4 aggregates prototypes whose ℓ_2-normalized embeddings lie within cosine radius ρ=0.18. We sweep $\rho \in \{0.14, 0.16, 0.18, 0.20, 0.22\}$ and compare against two alternative family-construction procedures—agglomerative single-link clustering and HDBSCAN with `min_cluster_size`= 3—at matched audit budget (52 families).

Table 42: Family-construction sweep. "Head purity" is the blinded-audit purity on the top-15 families; "Coverage" is the share of total operator uses explained by the top-37 head+mid families; "Over-merge" is the fraction of audited head families that pool two reference labels (`sort+sort_desc`, etc.); "Over-split" is the fraction of reference labels spread across ≥ 2 head families. Purity/coverage are reported at the same 52-family audit budget.

Family rule	*Head purity (%)*	*Coverage (%)*	*Over-merge (%)*	*Over-split (%)*	*MBPP-Comp acc. (%)*
$\rho = 0.14$ (tight)	91.7	78.3	1.4	12.1	51.8 ± 1.1
$\rho = 0.16$	90.9	83.6	1.9	8.3	52.7 ± 1.0
$\rho = \mathbf{0.18}$ *(ours)*	**90.5**	**87.4**	**2.3**	**5.1**	$\mathbf{53.4 \pm 1.0}$
$\rho = 0.20$	87.4	89.8	5.1	3.4	52.9 ± 1.1
$\rho = 0.22$ (loose)	82.6	92.1	9.7	2.2	51.3 ± 1.2
Agglom. single-link	85.1	85.3	6.8	6.7	51.9 ± 1.1
HDBSCAN (m=3)	86.9	82.7	4.3	7.9	51.1 ± 1.3

Tight radii preserve purity but over-split reusable families, while loose radii raise coverage by merging distinct behaviors. The selected ρ=0.18 operating point reaches 90.5% head purity, 87.4% coverage, the best MBPP-Comp accuracy, and the lowest joint over-merge/over-split profile among the tested rules. Agglom. single-link and HDBSCAN both lie behind this purity–coverage frontier.

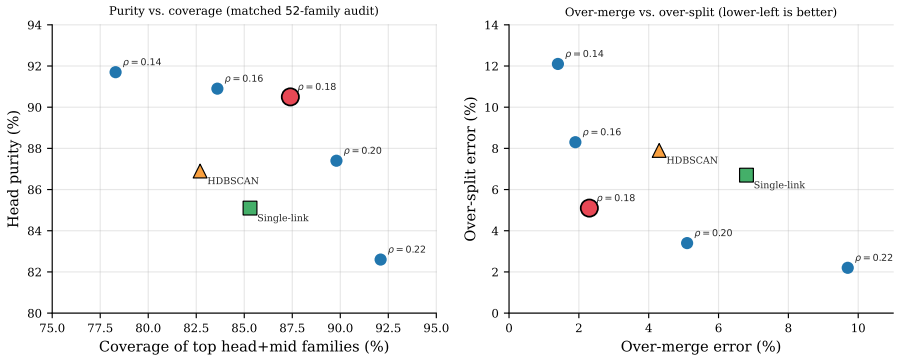

Figure 18: Head purity vs. coverage at matched audit budget (52 families) across $\rho \in \{0.14, 0.16, 0.18, 0.20, 0.22\}$ (blue circles) and two alternative family-construction rules (green square: single-link; orange triangle: HDBSCAN). The ρ=0.18 operating point (filled red) dominates both alternatives at purity $\geq 88\%$. Right panel: over-merge vs. over-split error, showing the expected trade-off; ρ=0.18 sits on the lower-left frontier.

Figure 18 places ρ=0.18 on the purity–coverage frontier and resolves errors into over-splitting and over-merging. Tighter rules fragment reusable families, looser rules merge distinct behaviors, and the selected radius balances both errors while reaching the highest MBPP-Comp accuracy among tested constructions.

Qualitative failure panel. Table 43 is the hand-curated companion to Tables 32 and 15, giving one concrete example per failure mode in repair and semantic-tail contexts.

Table 43: Representative observed failures in repair, semantic-tail annotation, and family construction.

Category	*Observed case*	*Consequence*
MBPP repair	Group-to-group-by-key swap inserts an extra keyword	Type check fails
ProofNet repair	Apply-to-symmetric-hypothesis swap leaves a side goal	Kernel rejects completion
MBPP semantic tail	Annotators split among map, reduce, and aggregate	Fleiss κ falls to 0.48
ProofNet semantic tail	Induction stub mixes induction, cases, and contradiction	Family label becomes multimodal
ρ=0.22 over-merge	Ascending and descending sort families fuse	Purity falls to 82.6%
ρ=0.14 over-split	Simplification family fragments across contexts	Coverage falls to 78.3%