# Same Text, Different Transitions: Action Insufficiency in Lean Provers

**Edison Yang**[1] **Neel Marripalapu**[2] **Akshaj Agadi**[3] **Andrew Kim**[1]
[1]Lambert High School [2]The University of Texas at Dallas [3]Texas A&M University
edisonly2200@gmail.com

## Abstract

Neural Lean provers usually give the model only the printed text of a proof state, and some searches reuse it to decide when two states are the same. Both uses assume that states printing identically behave identically under the same tactic. We show that this assumption fails. Searching Mathlib, we find 70 verified classes of identically printed states; in 68 of them, one tactic succeeds on one member and fails on another, yielding 103 contradictory cases verified by Lean. Controlled experiments trace the difference to elaborated structure that Lean's notation hides, and reproduce it on current Lean and Mathlib. A compact structural representation recovers the missing signal for mechanisms seen in training and fails to transfer to unseen ones. In a constructed benchmark with a known proof, merging states by identical text discards that proof in 48.4% of alias arrivals, while a structural identity prevents every loss at 8% runtime overhead. This rate is conditional on meeting an alias; in blind sampling, confirmed aliases arise at only 0.29% of search roots. Text written for people is thus not necessarily a safe identifier for machine state, and we suggest giving the model that text while identifying search states by structure.

## 1 Introduction

Neural Lean provers are given a textual rendering and asked to select a tactic. A pretty-printer is not injective, and by itself that is unremarkable: notation exists to suppress detail. The consequential question is whether the suppressed detail changes what a tactic does, because the same text is used as the policy's input and, in some systems, as the identity of a search node. We show that this interface can map action-distinct states to the same text. Table 1 is one instance: changing only the order of `intro h` and `simp` reaches a common 206-byte rendering, after which another `simp` fails on one state and succeeds on the other. The outcomes repeat on fresh sessions.

| | history | then `simp` |
|---|---|---|
| $s_1$ | `intro h; simp` | fails (×3) |
| $s_2$ | `simp; intro h` | succeeds (×3) |

Table 1: Two states in one Mathlib proof (`lintegral_ofReal_le_lintegral_nnnorm`), reached from a common ancestor by different tactic sequences. Both render to an identical 206-byte string; the same tactic, replayed three times per state on fresh sessions, deterministically fails on one and succeeds on the other. The hidden difference is the dominant coercion mechanism of §5: the two shallow explicit renderings are 1,167 and 1,183 bytes and carry `ENNReal.ofNNReal h` against `@WithTop.some.{0} NNReal h` in two positions, with equal environment digests and distinct structural digests.

Learned tactic generators for interactive theorem provers now close a substantial share of held-out library and competition problems (Bansal et al., 2019; Polu and Sutskever, 2020; Han et al., 2022; Lample et al., 2022; Yang et al., 2023; Jiang et al., 2023), on benchmarks such as miniF2F (Zheng et al., 2022) and PutnamBench (Tsoukalas et al., 2024). Many share one interface decision: the policy conditions on a printed rendering of the proof state, and some best-first search implementations also use that text as the identity of a search node (Yang et al., 2023; Lample et al., 2022). These uses assume that the observation identifies the state's response to an action, a transition-preservation condition required by safe state abstraction (Givan et al., 2003; Li et al., 2006). It is also implicit in learning tactic semantics from observed transitions (Samoylov and Vosoughi, 2025; Lamont et al., 2025). We test that assumption directly.

The issue also has a supervised-learning formulation. When the state rendering and tactic text are identical but compiler acceptance differs, an acceptance predictor given only text receives con-

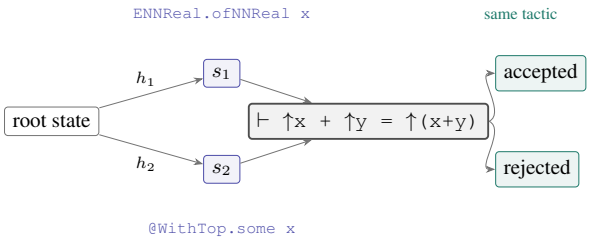



Figure 1: Observation aliasing. The point of interest lies in the two resulting states, which are indistinguishable to the policy; tactic order affecting an outcome is expected on its own. They differ only in elaborated structure the printer collapses, so the rendering, and therefore the entire input supplied to the evaluated state-only generators, is byte-identical, yet the same tactic is accepted on one and rejected on the other. The histories serve only to reach the two states; what matters is the pair of outcomes at a single observation. The goal shown is the compact constructed pair of §5, chosen for legibility; Table 1 is a natural instance of the same mechanism, whose rendering runs to 206 bytes.

tradictory deterministic labels. The resulting error cannot be removed by model scale or optimization while the observation is fixed. This affects tactic-effect prediction, reranking, transition modeling, imitation learning, dataset construction, and evaluation wherever the rendered state serves as a complete conditioning variable.

The hidden difference in Figure 1 lies in how each coercion is elaborated: `ENNReal.ofNNReal x` versus `@WithTop.some.{0} NNReal x`. They are definitionally equal and both print as `↑x`, but `simp`'s syntactic progress test distinguishes them. We call two such states *observation aliases*: this is state, or perceptual, aliasing (Whitehead and Ballard, 1991; Singh et al., 1994) instantiated in a deterministic and compiler-checkable setting, where the compiler verifies both sides of a contradictory label pair. Figure 2 organizes the evidence by logical implication and keeps incidence and consequence on their respective denominators. Its arrows ask distinct questions: whether aliases exist, whether the hidden difference changes an action, whether structural information explains and predicts the change, and whether an unsafe retention rule can discard a proof. Evidence for one arrow does not substitute for the denominator of another.

Over Mathlib (The mathlib Community, 2020) in Lean 4 (Moura and Ullrich, 2021) through Lean-

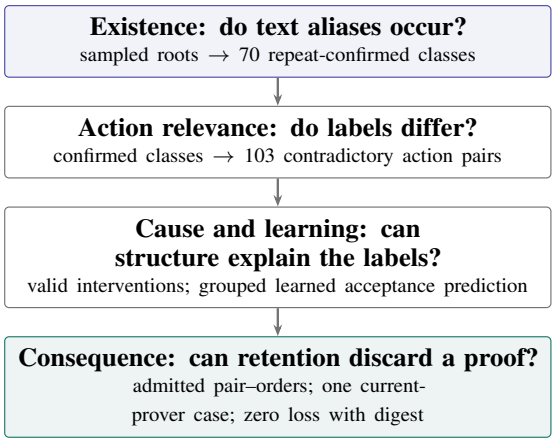



Figure 2: Conceptual evidence spine. Each arrow asks a separate question and uses the denominator printed beneath it. Population incidence, contradictory actions, structural learnability, and conditional search loss are estimated separately.

Dojo (Yang et al., 2023), we make three contributions:

1. **Impossibility.** We give repeat-verified counterexamples to action sufficiency: 68 natural alias classes contain opposite acceptance labels for the same tactic, yielding 103 contradictory textual inputs (§2–4). This conclusion is independent of the frequency with which a current policy encounters an alias.

2. **Empirical phenomenon.** Controlled interventions identify hidden elaborated structure as causal. A small acceptance model learns to use a compact structural channel, with theorem-grouped and mechanism-grouped evaluations separating interpolation from mechanism transfer (§5–7).

3. **Engineering consequence and repair.** A current step-prover and a 96-pair alias-conditioned stress benchmark demonstrate proof loss under unsafe retention. A structural identity key prevents every controlled loss at modest cost (§8).

Blind incidence, candidate exposure, and broad paired searches estimate the separate conditions under which the representational failure affects a search. Data, harnesses, and per-class outcomes are released.

## 2 Problem and scope

Let $T(s, a)$ denote the result of executing tactic $a$ at Lean tactic state $s$, and $\Psi$ the observable out-

come (acceptance with a successor, rejection, or proof closure). Rejection is what the interaction API reports as an error, including Lean's `made no progress`, the outcome most of the acceptance contradictions below take. A policy consumes $\varphi(s)$, the printed state; write $s_1 \sim_\varphi s_2$ when $\varphi(s_1) = \varphi(s_2)$.

The observation $\varphi$ is *action-sufficient* if $s_1 \sim_\varphi s_2$ implies $\Psi(T(s_1, a)) = \Psi(T(s_2, a))$ for all $a$. This is a necessary one-step transition-consistency condition for abstractions such as bisimulation quotients and MDP homomorphisms (Givan et al., 2003; Li et al., 2006), which impose still stronger preservation requirements. We test only this one-step condition; its failure rules the stronger properties out. We distinguish acceptance, visible successor, and bounded-continuation equivalence throughout.

**Acceptance ambiguity.** For alias class $C$, some $a$ has $r(s_i, a) \neq r(s_j, a)$, where $r \in \{0, 1\}$ is acceptance. The pair $(\varphi(s), a)$ then carries contradictory verified labels, and any acceptance predictor conditioned on it errs on at least $\min(n_0, n_1)/(n_0 + n_1)$ of members under the empirical member distribution. This lower bound applies to prediction of a specified action. The existence of a common successful tactic is a separate property.

Thus the corpus supplies contradictory supervision that the compiler certifies: Lean deterministically validates both labels on restored executable states, so each contradiction is a reproducible property of the data. A model given only the text can learn the average label distribution for an aliased input, while prediction specific to a member requires an additional observation channel.

Candidate exposure and a direct test on the model are reported in §7; the main text then evaluates search consequences.

**Implementation scope.** The observation claim is independent of a search implementation. The search-system claim is narrower: in the evaluated stack LeanDojo's `TacticState` compares and hashes on the rendering with the executable handle excluded (`compare=False`), and the best-first prover keys nodes on that value, so $\sim_\varphi$-equivalent nodes merge. LeanDojo-v2 uses Pantograph state handles and a history stack in its active prover, which avoids this particular merge. Its model prompt still uses the rendered state without the handle, so the action-sufficiency result continues to apply to policy conditioning. Appendix C gives the source and behavioral audits. A merge costs a proof when the discarded member's bounded continuation language contains a proof absent from the retained member's language.

## 3 Related work

**Proof-state interfaces.** LeanDojo (Yang et al., 2023) and LLMSTEP (Welleck and Saha, 2023) exemplify Lean interfaces that send a visible proof state to a language model. miniCTX (Hu et al., 2025) shows that file context beyond that state improves proving across problems. Our within-theorem members share the file, declaration, statement, and retrievable premises, so that context is identical. Earlier systems expose richer inputs: CoqGym's ASTactic encodes term trees (Yang and Deng, 2019), while TacTok combines proof-state semantics with partial proof-script history (First et al., 2020). A history-conditioned policy can distinguish the two arrivals studied here, although a merged node must still represent both members. LeanDojo, Pantograph (Aniva et al., 2025), and LeanTree (Kripner et al., 2025) define model or search interfaces without testing action sufficiency.

**Structural and transition-grounded representations.** Structural encodings include graph representations of higher-order logic (Paliwal et al., 2020) and Graph2Tac for Coq definitions (Blaauwbroek et al., 2024). Passport improves Coq proof synthesis with identifier categories, subwords, and paths (Sanchez-Stern et al., 2023). For Lean premise selection, Petrovčič et al. (2025) combine textual state embeddings with dependency graphs. These approaches enrich syntax, identifiers, or library context. Our paired members hold library context fixed and differ in elaborated structure local to the member. Samoylov and Vosoughi (2025) represent tactic semantics through induced state transformations, and 3D-Prover (Lamont et al., 2025) filters candidates by learned transition representations. Olejniczak et al. (2026) exploit proof-state symmetries. The present question is whether the visible state defines the transition being modeled. Contradictory deterministic transitions show that it need not. The same counterexample limits any tactic equivalence metric computed from text, such as LeanTutor's (Patel et al., 2025).

**Partial observability and search abstraction.** Goal-clone detection identifies alpha-equivalent Rocq goals to expose proof reuse opportuni-

| | blind | coerc. | struct. |
|---|---|---|---|
| roots replayed | 1,025 | 1,423 | 3,882 |
| expandable | 644 | 947 | 2,553 |
| with reconvergence | 112 | 234 | 667 |
| rendering-identical classes | 150 | 321 | 893 |
| flagged divergent | 11 | 50 | 36 |
| repeat-confirmed | **3** | **47** | **21** |
| new after deduplication | 3 | 47 | 20 |
| acceptance-ambiguous | 3 | 46 | 19 |
| contradictory pairs | 6 | 78 | 19 |

Table 2: Discovery by sampling regime. Only the blind column estimates incidence; targeted columns provide examples for mechanism analysis. One cross-regime duplicate leaves 70 unique classes.

ties (Ghanbari, 2025). Our operational criterion asks whether an equality used for policy input or node merging preserves action outcomes. Observation aliasing is the classical failure of a policy over observations to see its transition state (Whitehead and Ballard, 1991; Singh et al., 1994; Kaelbling et al., 1998). Tactic searches deduplicate states, the proof-search analogue of a transposition table (Zobrist, 1970); unsafe merging is the graph-history interaction problem (Kishimoto and Müller, 2004). Here elaboration creates the path dependence; game repetition rules are irrelevant.

## 4 Failure: search-reachable aliases

**Method.** From a root state we expand with a fixed tactic pool augmented by goal-derived rewrites and record, for every rendering, the set of *canonical* histories reaching it, where a step joins a history only if it changes the rendering (Appendix B). Classes reached by two prefix-free canonical histories are probed: one live state per history in a single session, the same battery for all members, and every flagged class re-verified on a fresh session with three repetitions per member. States are read at five levels of printer explicitness by metaprogram probes through the ordinary interaction API (Appendix B); derived digests separate the elaborated goal ($F_{\text{core}}$), the environment ($F_{\text{env}}$), and executable identity. Configuration: Mathlib `29dcec07`, Lean `4.10.0-rc1`, LeanDojo 2.0.2. The structural column combines two targeted sweeps. The second selected 2,400 roots and restored 2,234; expansion counts exclude failed restorations.

Table 2 separates incidence from discovery. In the blind sample, 112/644 expandable roots reconverge to an identical rendering, but only 3 classes on 3/1,025 sampled roots are confirmed operationally divergent: 0.29% (95% CI 0.06–0.85%). The targeted regimes contribute 67 more unique classes and support mechanism analysis only. In total, the 70 classes span 53 theorems, 67 roots, and 40 files in nine top-level Mathlib directories. A corpus-wide cross-file census is dominated by declaration context and is reported separately (Appendix A).

Human scripts rarely revisit identical renderings; the aliases studied here arise primarily through search reconvergence (Appendix A).

**Acceptance is contradictory.** In 68 of 70 classes some tactic is accepted on one member and rejected on another, stably across three repetitions on fresh sessions; the remaining two diverge only in successor states. This yields 103 contradictory observation–action pairs: identical model input, identical tactic text, opposite verified acceptance. Any acceptance predictor conditioned on $(\varphi(s), a)$ errs on at least $\min(n_0, n_1)/(n_0+n_1)$ of members per pair. Over the 103 pairs the member-weighted aggregate is 0.40, a conditional inconsistency floor restricted to these pairs. The phenomenon spans several elaboration mechanisms, whose distributions are reported below.

## 5 Cause: hidden elaborated structure

Reading each class at five levels of printer explicitness localizes the hidden content (Appendix B). The consumed rendering $\varphi_{\text{state}}$ and its depth-expanded variant fail to separate any class, and the environment digest $F_{\text{env}}$ also fails wherever the probe completes. This excludes ambient declaration context as the cause. Disabling notation under `pp.all` separates 67 of 70, showing that the notation layer collapses most of the hidden content (Ullrich and de Moura, 2020); depth elision has no effect. The hidden content falls into six named categories plus two opaque classes: `ENNReal` coercion (56 classes), `Fin` eta-expansion (4), local-definition unfolding (4), instance paths (2), `Ne` versus `Not (Eq ...)` (1), and a proof term difference (1). In the dominant family the same token pair separates the members every time: a sixteen-byte difference in 48 of the 53 classes where the rung is measurable, and a larger one where the token recurs. The divergence follows from tactic implementation: `simp`'s progress check compares the resulting term syntactically, so one call can rewrite one member and report no progress

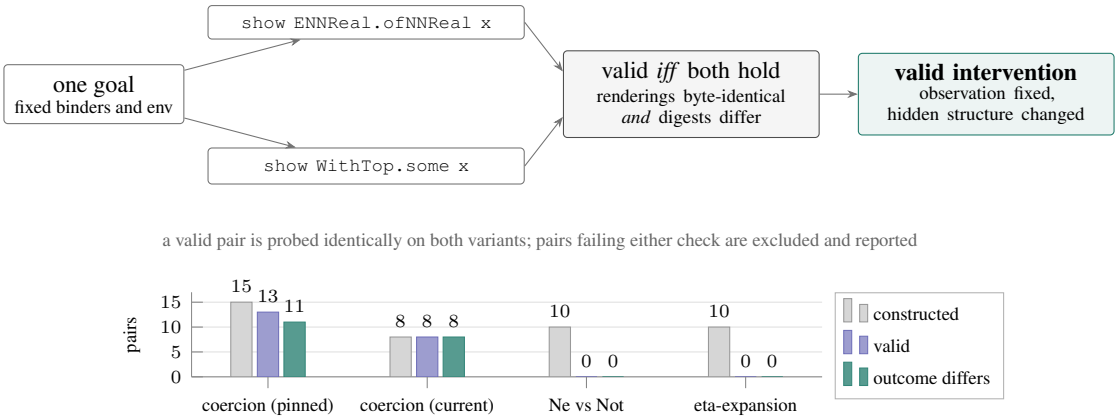



Figure 3: Rendering-preserving interventions. **Top:** the construction. A single `show` step replaces the goal with a definitionally equal term, so only the elaborated representation changes; a pair is used only if it passes both checks. **Bottom:** outcomes. The construction succeeds for the coercion family under the pinned stack and, with acceptance-sensitive probes, on a current Lean and Mathlib, where all 8 pairs are accepted on one variant and rejected on the other. The other two families yield zero valid pairs because the printer renders their two spellings differently.

on the other. Its direction depends on the simp set. A lemma stated for the abbreviation, as in the interventions below, rewrites only the member holding it. The default set rewrites the unfolded form: on the Table 1 witness, `simp` succeeds on the unfolded member with a successor that renders identically, and makes no progress on the member already holding the abbreviation.

**Rendering-preserving interventions.** The classes above are observational. To manipulate the hidden structure directly we construct pairs of standalone examples (Figure 3) identical in every respect except a single `show` step replacing the goal with a definitionally equal term carrying different hidden syntax: same environment, binders, goal statement, and probe text. `show` accepts any term definitionally equal to the goal, so it alters the elaborated representation without altering what is proved. Validity is verified per pair in both directions: the intervention must have taken effect, so the variants' structural digests must differ, and it must have remained hidden, so their pre-probe renderings must be byte-identical. Pairs failing either check are excluded and reported.

For the dominant coercion family 13 of 15 pairs pass both gates, and 11 of 13 change the probe outcome. The attempted `Ne` and eta interventions fail the byte-identity gate because their source spellings remain visible. The intervention therefore establishes a causal mechanism for the dominant family; the other observed families remain observational.

**Visibility under continuation.** In the original 59-class corpus, a shared continuation makes 18 classes visibly different, including 17 at `contrapose!`. A reduced battery that omits this operation yields near-total confinement. Escape is therefore a property of both the state and the tactic distribution (Appendix A).

**Current stack.** We repeat blind discovery on Lean `v4.33.0-rc1` and contemporaneous Mathlib. Sixty uniformly sampled compiled files yield 2,507 states. From 600 fixed-seed roots, 570 restore successfully and 98 rendering-alias classes are found. Fresh repeated probing confirms two classes at one root, with seven contradictory class–action pairs. The resulting root incidence is 1/570, or 0.18% (Clopper–Pearson 95% CI 0.004–0.97%). Both classes occur in `ModuleCat.disjoint_span_sum`; explicit printing separates terms produced by `rw` and `simp only`. In addition, all eight current coercion interventions pass both validity gates and reproduce opposite probe acceptance. Historical script replay reproduces one of eight older witnesses; the remaining seven fail after one lemma rename and six cases of API drift (Appendix C).

## 6 Learning from a structural channel

The observation aliases establish that the default rendering omits information relevant to the transition. We next train a deliberately small model to predict tactic acceptance from a compact structural channel local to the member. This test asks whether

a learned predictor can use information unavailable in the rendering. Candidate exposure and zero-shot use by a frozen prover are evaluated separately in §7.

**Representation.** For each restored member we extract a sorted multiset of application-head constant paths from a shallow explicit rendering of the local context and goal. This *structural sketch* restricts its features to structural constants. Tactics, pair identities, histories, theorem names, and outcome labels are excluded. The sketch compresses the explicit rendering before it is given to the evaluator.

**Learned acceptance model.** The extractor succeeds for all 178 members of the 70 confirmed classes. The acceptance evaluator selects one accepted and one rejected member for each repeat-confirmed class–action contradiction, giving 206 examples in 103 pairs from 51 theorems. We fit logistic regression over explicit sketch features and action–feature crosses ($L_2$=$10^{-3}$, 500 gradient steps). Each split constructs an explicit sparse vocabulary from its training examples alone. Coordinates first observed in the test split are omitted and receive zero weight. The theorem-grouped protocol keeps every example from one theorem in a single fold. It estimates transfer to unseen theorems while allowing mechanisms to recur across folds. The mechanism-grouped protocol defines groups from the hidden elaborated delta: coercion construction, `Fin` eta-expansion, instance path, local-definition unfolding, and the residual printer-blind classes. Each group is held out in turn. The mechanism macro weights the four audited mechanisms equally; the all-group macro also includes the residual group. The `Ne`-versus-`Not` and proof-term classes yield different successful successors without opposite acceptance labels, so they fall outside this binary evaluation. The paired metric assigns 1 when the accepted member receives the higher score, $1/2$ for a tie, and 0 for a reversal. Confidence intervals use the theorem as the resampling unit.

**Within-mechanism generalization.** Under theorem-grouped evaluation, the shallow sketch correctly orders 97 pairs, ties 5, and reverses 1, giving a paired rank of .966 (bootstrap over theorem clusters, 95% CI .927–.995). All 78 coercion pairs are correctly ordered. The default rendering remains constant within every pair and therefore scores .500. The structural digest separates all classes and serves only as an opaque identity key.

| Representation | classes split | pair rank | accuracy |
|---|---|---|---|
| default rendering | 0/70 | .500 | .500 |
| structural sketch | 67/70 | **.966** | .850 |
| structural digest | 70/70 | n/a | n/a |

Table 3: Theorem-grouped learned acceptance prediction on 103 contradictory pairs. Pair rank measures whether the accepting member is scored above the rejecting member. The digest is excluded from model features.

| Held-out group | classes | pairs | pair rank |
|---|---|---|---|
| `Fin` eta | 4 | 10 | 1.000 |
| instance path | 2 | 7 | .857 |
| coercion | 56 | 78 | .500 |
| local unfolding | 4 | 4 | .625 |
| residual | 2 | 4 | .500 |
| mechanism macro | | | .746 |
| all-group macro | | | .696 |
| pair weighted | | | .578 |

Table 4: Leave-one-group-out transfer with a fold-local explicit vocabulary. Each row trains on all other groups. The mechanism macro weights the four audited mechanisms equally; the all-group macro adds the residual group. The pair-weighted score retains the corpus distribution.

**Mechanism transfer.** The recurring coercion mechanism contributes 56 of 68 evaluated classes and 78 of 103 pairs. When coercion is held out, none of its 78 pairs contains a member-distinguishing coordinate observed in the training mechanisms; all 78 pairs therefore tie. The held-out `Fin` and instance-path groups score 1.000 and .857, and all 17 pairs contain a distinguishing coordinate observed during training. Local unfolding scores .625; three of its four pairs have such a coordinate. These results localize transfer to mechanisms that share informative structural coordinates with the training groups.

The explicit map removes collision and dimension choices. Across $L_2 \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, theorem-grouped rank ranges from .956 to .966 and the pair-weighted mechanism holdout ranges from .558 to .578. The mechanism macro ranges from .621 to .746, chiefly because the four-pair local unfolding group ranges from .125 to .625. The theorem-grouped result supports interpolation across unseen theorems within observed mechanisms. The mechanism-held-out analysis establishes partial transfer for two small audited mecha-

| Input | all | coerc. | `Fin` | inst. | unfold | macro |
|---|---|---|---|---|---|---|
| ordinary rendering | .500 | .500 | .500 | .500 | .500 | .500 |
| + structural sketch | .539 | .564 | .400 | .429 | .625 | .504 |

Table 5: Frozen-prover paired ranking on 103 confirmed tactics. Each entry is the fraction for which the accepting member scores above the rejecting member, with ties assigned one half. The macro weights the four audited groups of hidden deltas equally.

nisms and chance performance for the dominant unseen mechanism. Broader generalization requires training evidence that spans additional hidden structural deltas.

Across all 178 members, the median sketch is 934 bytes (maximum 2,503), compared with 4,212 bytes (maximum 282,108) for shallow `pp.all`. Shallow sketches distinguish 67 of 70 classes and the digest distinguishes all 70. The learned model therefore establishes compact identifiability of the hidden signal within observed mechanisms. The prototype currently obtains an explicit rendering before compression, leaving direct extraction as an engineering problem.

## 7 Direct evaluation on a frozen prover

The learned evaluator in §6 uses information in the sketch after training on the task. We next test whether a current prover can use that channel without adaptation, using the same paired acceptance task.

**Protocol.** We freeze BFS-Prover-V2-7B (Xin et al., 2025) at one immutable snapshot. For each of the 103 repeat-confirmed class–action contradictions, the model scores the same tactic on one accepting and one rejecting member. The ordinary arm uses the model-card prompt `{state}:::`. The structural arm places the sketch for that member in a Lean comment between the rendering and the separator. We average the conditional log probabilities of the tactic tokens and count whether the accepting member receives the higher score; ties receive one half. The checkpoint, prompt, and scoring rule remain fixed. No outcome label, theorem, class, or mechanism from the alias corpus enters experiment-side training or calibration. We report every group of hidden deltas separately and macro-average the four audited groups. Confidence intervals resample the 51 theorems.

**Result.** The structural input obtains .539 paired accuracy (bootstrap over theorem clusters, 95% CI .440–.644), with 53 correct rankings, 5 ties, and 45 reversals. Its macro average across the four audited mechanisms is .504. The ten pairs with equal augmented-prompt token counts score .550. The ordinary arm ties exactly on every pair because the rendering and tactic are byte-identical. Appending the sketch to this frozen prover provides no reliable acceptance-sensitive signal. Together with the trained model's .966 theorem-grouped score, this result decomposes structural-channel utility into information availability and model adaptation. The experiment conditions on a confirmed tactic; candidate generation and proof rate at the theorem level remain outside its estimand.

| $k$ | mean $\lvert A\rvert$ | member cov. | ambiguous | headroom |
|---|---|---|---|---|
| 1 | 1.0 | .376 | 1 | .000 |
| 2 | 2.0 | .798 | 2 | .000 |
| 4 | 4.0 | .882 | 2 | .000 |
| 8 | 7.5 | .938 | 2 | .000 |
| 16 | 11.9 | .938 | 2 | .000 |

Table 6: Candidate exposure over nested miniCTX beam prefixes. "Ambiguous" counts classes with an acceptance-divergent candidate. Headroom is the member-aware oracle coverage minus the best shared-action coverage. All 6,354 compiler calls agree within each state–candidate unit.

**Candidate exposure.** One deterministic miniCTX (Hu et al., 2025) beam-16 list is generated for each of the 70 class renderings, with a 16-token tactic cap. Nested prefixes give $k = 1, 2, 4, 8, 16$, and every candidate is executed three times on all 178 members. Table 6 shows that wider prefixes improve general coverage through $k = 8$, while acceptance-divergent exposure remains confined to two classes. The member-aware and best shared-action oracles have identical coverage at every width, giving zero member-specific headroom. Thus the structural channel identifies member differences that the present candidate sets do not convert into a distinct available proof choice. Fixed beam-8 results on the original corpus for miniCTX and two additional generators are reported in Appendix E.

## 8 Consequence: search and retention

Aliasing matters to search only when a policy reaches a class and continuation quality differs across its members. We evaluate conditional loss after an alias encounter, natural continuation loss, and broad policy incidence separately.

| | budget | | | |
|---|---|---|---|---|
| Proposal generator | 2 | 4 | 8 | 12 |
| *pairs proved by the refined arm (of 96)* | | | | |
| targeted | 96 | 96 | 96 | 96 |
| context bank | 36 | 72 | 96 | 96 |
| generic first | 96 | 96 | 96 | 96 |
| *rendering loss, % of covered pair–orders* | | | | |
| targeted | 48.4 | 48.4 | 48.4 | 48.4 |
| context bank | 50.0 | 50.0 | 50.0 | 48.4 |
| generic first | 31.3 | 31.3 | 31.3 | 31.3 |

Table 7: Exact results under uniform order on 96 admitted stress pairs. Loss is conditioned on the refined arm exposing a proof within budget. For the targeted generator, the pair-clustered 95% bootstrap interval is 46.4–50.0%; the eight-base-template interval is 46.9–49.5%.

**Representative sensitivity.** Every member in the original 59-class corpus is restored and searched independently under the same budget, model, and seed. With BFS-Prover-V2-7B (Xin et al., 2025), the first candidate list is byte-identical in all 59 classes; tactic execution alone then creates different trajectories. A proof is found in only two classes, and in only one do the members' outcomes differ. In `snormEssSup_add_le`, the `intro h; simp` member is proved at expansion 14; the `simp; intro h` member remains unproved across repetitions and at triple budget. A rendering-keyed merge can therefore discard the member proved under this continuation. An ancestor-rooted control proves through another intermediate under either retention order, localizing the observed loss to the aliased node (Appendix D).

**Alias-conditioned stress benchmark.** We cross eight current-stack, rendering-preserving intervention pairs with 12 proposition contexts. Admission requires Lean to verify byte-identical default renderings, distinct structural digests, and a fixed continuation that closes one member and fails on the other. All 96 pairs pass. After both branches reach the alias, rendering identity retains the first arrival; rendering plus digest retains both. We enumerate both arrival orders exactly and also run 200 seeded random orders per pair under three proposal generators and four budgets.

The targeted generator exposes the known continuation and `rfl`. The refined arm proves all 96 pairs under both orders. Rendering identity loses the proof for 93 of 192 pair–order combinations, giving the exact probability under uniform order $93/(2 \cdot 96) = 48.4\%$; the randomized estimate at budget 2 is 48.0%. Table 7 shows that context-bank and generic-first generators change the conditional rate. This variation identifies proposal exposure as part of the estimand. The benchmark establishes a conditional failure of bounded-search completeness under controlled alias encounters. Its shared coercion mechanism prevents a population-level interpretation (Appendix D).

**Broad nulls.** Paired searches with rendering-only versus structurally refined identity prove the same theorem in all 158 runs (36 proved in both; one-sided 95% upper bound 1.9% for discordance). Only 4 runs actually merge fingerprint-distinct states. Probing all 25 such merge pairs found no operational disagreement (upper bound 11.3%). Separately, replaying the known remainder of a Mathlib proof on 113 aliases loses none (upper bound 2.6%). These results constrain the broad rate under the tested policies and budgets. The 7B member search on the original corpus is outcome-informative in 2 of 59 classes and loses one local continuation. Policy exposure and candidate set regret are reported in §7 and Appendix E.

**Identity refinement.** Model observation and node identity can differ. A 64-bit hash of the goal closed over its context and normalized for metavariables separates all 70 classes and prevents every stress-benchmark loss. In a rotated 24-theorem cost test, verbose printing raises end-to-end time by 179%, versus 8% for the digest (Appendices B and C). The digest is a prototype: it is specific to one Lean version, can collide in principle, and provides no proof of operational equivalence. It serves as an opaque search identity, while the structural sketch in §6 is a feature for the model.

## 9 Implications for prover design

**Separate interfaces.** Pretty printing serves human communication and policy conditioning. Deduplication requires an identity with stronger transition-consistency properties. The results support separate representations for these functions: a stable textual observation for the policy and a structurally refined key for the search graph. Version tagging and collision checks remain necessary for a production identity scheme.

**Structural conditioning.** The paired-ranking result motivates a compact structural channel alongside the rendering. The sketch is local to the mem-

ber, generalizes across unseen theorems within observed mechanisms, and is substantially smaller than shallow `pp.all`. The trained acceptance model and the frozen prover intervention together locate the present bottleneck: the signal is learnable under training on the task, while an unadapted prover gives near-chance action scores. Direct extraction from Lean expressions could remove the prototype's explicit-printing stage. The digest remains reserved for opaque search identity.

**Candidate generation.** Member-specific scoring can affect proof search when the candidate set contains a corresponding choice. Increasing miniCTX beam width raises shared-action coverage but produces no member-specific oracle headroom through $k = 16$. Structurally conditioned generation or targeted proposal mechanisms are therefore the next experimental step. Candidate exposure and candidate set regret should accompany top-$k$ accuracy when evaluating such systems.

## 10 Conclusion

Hidden elaborated structure produces contradictory deterministic labels for identically rendered state–tactic inputs, disproving action sufficiency for the standard textual observation. A compact structural channel makes the missing signal learnable within observed mechanisms, and controlled retention experiments show that rendering identity can discard an available proof while a structural key preserves it at modest cost. Human-readable rendering and machine-state identity serve distinct design objectives. Using one in place of the other fails transition safety; the demonstrated failure is removed by structurally refined identity.

## Limitations

**Representation and mechanism.** The central claim concerns action sufficiency, for which one verified counterexample is decisive. The 70-class corpus characterizes the phenomenon under the reported search protocol and remains concentrated: 56 classes involve coercion construction, with four `Fin` eta, four local-unfolding, two instance-path, two residual, one `Ne`-versus-`Not`, and one proof term class. Causal identification applies to the dominant family. Theorem-grouped prediction measures interpolation within observed mechanisms. Mechanism-held-out transfer depends on shared training coordinates, and the dominant unseen coercion group ties at chance.

**Models and search.** The test on the frozen prover evaluates one prompt and scores confirmed tactics, and it is underpowered: its interval spans the null, so it bounds a small zero-shot effect without excluding one. Its scope is this checkpoint and this prompt, which deliver no usable signal specific to a member. Whether another frozen model would remains untested. The checkpoint may have encountered Mathlib during training. Candidate-set analyses measure exposure within fixed decoded lists, while proof-rate effects also depend on generation, search policy, and budget. The stress benchmark estimates conditional completeness loss after an admitted alias encounter with an available member-separating continuation. Its 96 pairs derive from one mechanism crossed with logical contexts. Blind sampling and broad paired searches estimate encounter frequency under the evaluated distributions; the single natural loss concerns the aliased node's bounded continuation.

**Scope beyond this stack.** Every measurement here is confined to Lean 4, Mathlib, and the LeanDojo interface. The two preconditions are general, namely a printer that suppresses elaborated distinctions and an interface that exposes only its output, and other assistants and their neural interfaces satisfy them in outline. We have not measured any of them, so whether comparable aliases arise in Rocq, Isabelle, or non-LeanDojo Lean interfaces remains untested.

## References


Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, and Sanmi Koyejo. 2025. Pantograph: A machine-to-machine interaction interface for advanced theorem proving, high level reasoning, and data extraction in Lean 4. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. 2019. HOList: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning (ICML)*, pages 454–463.

Lasse Blaauwbroek, Miroslav Olšák, Jason Rute, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. 2024. Graph2Tac: Online representation learning of formal math concepts. In *International Conference on Machine Learning (ICML)*.

Emily First, Yuriy Brun, and Arjun Guha. 2020. TacTok: Semantics-aware proof synthesis. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):231:1–231:31.

Ali Ghanbari. 2025. Automatic goal clone detection in Rocq. In *39th European Conference on Object-Oriented Programming (ECOOP)*, volume 333 of *Leibniz International Proceedings in Informatics*, pages 12:1–12:19.

Robert Givan, Thomas Dean, and Matthew Greig. 2003. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. 2022. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations (ICLR)*.

Jiewen Hu, Thomas Zhu, and Sean Welleck. 2025. miniCTX: Neural theorem proving with (long-)contexts. In *International Conference on Learning Representations (ICLR)*.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations (ICLR)*.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.

Akihiro Kishimoto and Martin Müller. 2004. A general solution to the graph history interaction problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 644–649.

Matěj Kripner, Michal Šustr, and Milan Straka. 2025. LeanTree: Accelerating white-box proof search with factorized states in Lean 4. *arXiv preprint arXiv:2507.14722*.

Sean Lamont, Christian Walder, Amir Dezfouli, Paul Montague, and Michael Norrish. 2025. 3D-Prover: Diversity driven theorem proving with determinantal point processes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. 2022. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. 2006. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

Jannis Limperg and Asta Halkjær From. 2023. Aesop: White-box best-first proof search for Lean. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*.

Leonardo de Moura and Sebastian Ullrich. 2021. The Lean 4 theorem prover and programming language. In *Automated Deduction (CADE)*.

Krzysztof Olejniczak, Radoslav Dimitrov, Xingyue Huang, Bernardo Cuenca Grau, Jinwoo Kim, and İsmail İlkan Ceylan. 2026. What are the right symmetries for formal theorem proving? *arXiv preprint arXiv:2605.22257*.

Aditya Paliwal, Sarah Loos, Markus Rabe, Kshitij Bansal, and Christian Szegedy. 2020. Graph representations for higher-order logic and theorem proving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2967–2974.

Manooshree Patel, Rayna Bhattacharyya, Thomas Lu, Arnav Mehta, Niels Voss, Narges Norouzi, and Gireeja Ranade. 2025. LeanTutor: Towards a verified AI mathematical proof tutor. *arXiv preprint arXiv:2506.08321*.

Job Petrovčič, David E. Narváez, and Ljupčo Todorovski. 2025. Combining textual and structural information for premise selection in Lean. In *NeurIPS Workshop on Mathematical Reasoning and AI (MATH-AI)*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Elisaveta Samoylov and Soroush Vosoughi. 2025. Modeling tactics as operators: Effect-grounded representations for Lean theorem proving. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing (MathNLP)*, pages 168–175.

Alex Sanchez-Stern, Emily First, Timothy Zhou, Zhanna Kaufman, Yuriy Brun, and Talia Ringer. 2023. Passport: Improving automated formal verification using identifiers. *ACM Transactions on Programming Languages and Systems*, 45(2):12:1–12:30.

Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. 1994. Learning without state-estimation in partially observable Markovian decision processes. In *International Conference on Machine Learning (ICML)*, pages 284–292.

The mathlib Community. 2020. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*.

George Tsoukalas, Jasper Lee, John Jennings, Sangeetha Xin, Michelle Ding, Michael Honerkamp, Kyunghoon Kim, and Swarat Chaudhuri. 2024. PutnamBench: Evaluating neural theorem-provers on the Putnam mathematical competition. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Sebastian Ullrich and Leonardo de Moura. 2020. Beyond notations: Hygienic macro expansion for theorem proving languages. In *Automated Reasoning (IJCAR)*, pages 167–182.

Sean Welleck and Rahul Saha. 2023. LLMSTEP: LLM proofstep suggestions in Lean. In *NeurIPS Workshop on Mathematical Reasoning and AI (MATH-AI)*.

Steven D. Whitehead and Dana H. Ballard. 1991. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Ran Xin, Zeyu Zheng, Yanchen Nie, Kun Yuan, and Xia Xiao. 2025. Scaling up multi-turn off-policy RL and multi-agent tree search for LLM step-provers. *arXiv preprint arXiv:2509.06493*.

Kaiyu Yang and Jia Deng. 2019. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning (ICML)*, pages 6984–6994.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. miniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*.

Albert L. Zobrist. 1970. A new hashing method with application for game playing. Technical Report 88, Computer Sciences Department, University of Wisconsin. Reprinted in *ICCA Journal*, 13(2):69–73, 1990.

## A Corpus context and natural occurrence

**Cross-file census.** Across 168,074 auditable Mathlib states, 271 observation-collision classes diverge operationally, but approximately 99% differ only in declaration context: the members occupy different files or positions, so their environments differ. State-weighted prevalence on elaborated-goal identity is 0.03%. An intervention control adding valid declarations while holding the elaborated goal and tactic fixed produced one success-to-failure transition in 497 trials. Cross-file classes cannot be merged by a transposition table, since two theorems never co-occur in one search tree, which is why the main text measures within-search aliasing.

**Human proof trajectories.** Across 60,665 traced Mathlib proofs and 168,074 states, only 10 proofs revisit a byte-identical rendering, yielding 12 position pairs. In all 12 the author applies a different tactic at the two positions, leaving zero natural same-tactic pairs for probing. Linear proof scripts thus rarely create the repeated observation that search produces through reconvergence.

**Visible escape.** Escape depth is the first step at which member renderings or terminal outcomes differ under a shared continuation, generated by the policy from the shared observation and applied to every member in lockstep to depth 8, stopping at the first divergence, terminal disagreement, or revisited rendering. Of 59 classes, 19 escape and 18 pass repeat controls. Successor diffs are identical in form: one member's goal becomes $\forall b,\ h < b \rightarrow \ldots$ while the other's becomes $\neg\exists b,\ \cdots \wedge b \leq h$.

## B Measurement apparatus and protocol

**Fingerprint ladder.** One state is read at five levels of printer explicitness by `run_tac` probes: the default rendering $\varphi_{\text{state}}$; $\varphi_{\text{shallow}}$ (`pp.all`, notation disabled); $\varphi_{\text{deep}}$ (`pp.deepTerms`); $\varphi_{\text{proofs}}$; and $\varphi_{\text{full}}$. Two implementation properties matter for reproduction: `pp.all` leaves `pp.deepTerms` and proof-term suppression separately configurable, and module identity must be excluded from executable digests or every cross-file pair separates trivially.

**Experimental specification.** Policies: `leandojo-lean4-tacgen-byt5-small` and `ntp-mathlib-st-deepseek-coder-1.3b`, state-only input, inputs truncated at 2,300 tokens, deterministic beam search (8 beams), $k$=8; representative sensitivity additionally uses `BFS-Prover-V2-7B` (prompt `{state}:::`, deterministic beam-8, 24 expansions, 300 s per member, verification at 72 expansions and 900 s). Expansion pool: `simp`, `constructor`, `intro h`, `omega`, plus per-hypothesis `rw [h]` and `simp only [h]` ($\leq$16 per state), depth 2. Probe battery: `rfl`, `assumption`, `omega`, `simp`, `norm_num`, `constructor`, `exact?`, `trivial`, `decide`, `simp_all`, `tauto`, `aesop`, plus goal-derived rewrites; the merge-pair battery adds `contrapose!`, `push_neg`, `norm_cast`. Per-probe watchdogs 30–120 s; session timeouts 300–900 s; three repetitions per member; fixed seeds; shards deduplicated by (file, theorem, tactic index) and confirmed classes by (theorem, rendering hash).

For the direct test on the model, the frozen BFS-Prover-V2 checkpoint scores each confirmed tactic by mean conditional token log probability. The structural prompt appends that member's shallow sketch in a Lean comment before the model-card separator. The ordinary and structural arms use identical tactic tokens. The paired metric assigns

one, one half, or zero according to whether the accepting member scores above, ties, or scores below the rejecting member. The bootstrap over theorem clusters uses 10,000 resamples.

The width sweep uses miniCTX snapshot `7c00b1175a82e`, one deterministic 16-beam decode per class, and a 16-token cap. Prefixes of the single ranked list define each $k$, ensuring nested candidate sets. All candidates are executed three times per member; a failure includes parser, elaboration, and no-progress errors.

**Controls against false positives.** Three controls address hazards that produced false positives during development. After any watchdog interrupt the session is discarded, because channel desynchronization can return an earlier probe's answer. Every divergence must be self-consistent over three repetitions per member in a fresh session; this removed 11 of 13 candidates supported only by nondeterministic `aesop` (Limperg and From, 2023). Escape is reported relative to the tactic battery, and canonicalization based on the rendering was compared with a separating fingerprint on 369 paired roots. The artifact retains exclusions, repeat tables, and resource failure verdicts.

**Structural digest.** One 322-byte `run_tac` probe: `mkForallFVars (lctx.getFVars) type` closes the goal over its local context, converting free variables to de Bruijn indices; `instantiateMVars` resolves assignments; `abstractMVars` canonically renumbers remaining expression and universe metavariables; the resulting term's 64-bit `Expr.hash` is returned. Validated with three repetitions per member on all 59 classes in the original corpus (59/59 separated, all stable) and as a search key on 39 theorems. Table 3 extends the separation audit to the full 70-class corpus, where the digest separates 70/70 classes. The hash approximates normalized structural equality: collisions are possible in principle and the encoding is Lean-version specific.

## C Node identity and persistence across versions

**Version persistence.** The blind audit on the current stack samples 60 of 8,302 compiled files uniformly with a fixed seed and traces all 60; they contain 989 theorems and 2,507 tactic states. Of the 600 sampled roots, 570 complete restoration and expansion, while 30 resource or restoration failures are recorded and excluded from the incidence denominator. The two confirmed classes have two- and four-member histories differing through `rw [huv]` and `simp only [huv]`; shallow `pp.all` and the structural digest distinguish both. Seven probes give stable opposite acceptance labels and no probe is nondeterministic.

Eight representative classes were replayed in place on Mathlib master `3b3cdbb` under Lean `v4.33.0-rc1`, splicing each proof back into its own file so namespace and variable context are preserved. One reproduces fully, with both histories elaborating, all four members' renderings byte-identical, and the probe's successors still diverging. Of the remaining seven, one is a pure rename: the Table 1 witness persists as `lintegral_ofReal_le_lintegral_enorm` with its prefix referencing a renamed lemma, and six are blocked by tactic-level API drift. Every class that remained constructible retained the divergence.

**Node identity across package versions.** In the evaluated stack (LeanDojo 2.0.2) `TacticState` is a frozen dataclass whose printed field participates in equality and hashing while the executable handle is excluded with `compare=False`, and the shipped best-first prover indexes nodes by that value. Directly constructed states with one rendering and different handles compare equal, hash equal, and collide in a dictionary.

We separately audit the published LeanDojo-v2 1.0.9 wheel (SHA-256 `635e8f99...`) and its active Pantograph backend at commit `f8aee320`. The active `BaseProver` keeps a history stack and contains no state-indexed transposition dictionary. Pantograph `GoalState` includes `state_id`; two constructed states with equal rendered goals and different identifiers render equally and compare unequally. They are also unhashable because their goal fields are lists. Thus the active v2 prover preserves these distinct live handles and avoids the evaluated rendering-keyed merge.

The v2 wheel also contains a legacy search tree module whose `InternalNode` compares a `TacticState`, but the wheel omits the interaction state implementation that module imports, so we base the conclusion on the active Pantograph path, which supplies the model with `str(GoalState)`. Because that string omits `state_id`, equal renderings can remain observation aliases for the policy. The search loss claim ap-

| Identity | nodes | peak queue | time | probe share |
|---|---|---|---|---|
| rendering | 230 | 135 | 1.00× | 0% |
| shallow `pp.all` | 233 | 138 | 2.79× | 64% |
| structural digest | 232 | 136 | 1.08× | 15% |

Table 8: Node-identity cost over 24 rotated-order theorem searches. Time is total wall clock relative to the rendering baseline. All arms prove the same 4 theorems; node and queue differences occur on 2 of 24.

plies to the evaluated LeanDojo prover; the action sufficiency claim applies to both textual interfaces.

**Cost of a refined identity key.** The cost test runs twenty-four theorems at 12 expansions, rotates arm order with the theorem index, and adds a baseline arm that omits the probe. The baseline is needed because the paired-search harness computes $\varphi_{\text{shallow}}$ in every arm for instrumentation, so its recorded wall clock cannot isolate the key's cost. Node and queue counts appear in Table 8; each refined key differs from the baseline on 2 of 24 theorems. Expansion counts are nearly identical (150, 150, 151) because the budget caps them, so the counts are uninformative about node overhead. The printing probe costs 0.47 s per node at the median and consumes 64% of its arm's search time (paired median 2.55×, maximum 12.1×); the digest costs 0.090 s and consumes 15% (paired median 1.27×). Generation and tactic execution are unchanged across arms and all three arms prove the same 4 theorems.

## D Retention protocol and reproducibility

**Stress construction and admission.** Eight acceptance-divergent interventions on Mathlib `3b3cdbb`/Lean `v4.33.0-rc1` are crossed with 12 proposition contexts, including conjunction, disjunction, equality with `True`, double negation, and duplicated goals. Each context has a logical introduction sequence that exposes the base pair's acceptance-sensitive probe as the proof-producing step. Lean compilation then enforces three admission conditions: byte-identical default renderings, distinct structural digests, and closure of the abbreviated member with rejection on the unfolded member. All 96 candidates pass. The harness compiles 192 capture declarations and a matrix of 17,472 state–action labels. Capture declarations log the state before an explicit `sorry`; no state–action closure label relies on `sorry`.

**Retention and proposal protocol.** The comparison begins after both branches reach the admitted alias. Rendering identity retains the first arrival, while rendering plus digest retains both. The targeted generator emits the pair's known continuation followed by `rfl`. The context-bank generator emits the eight continuations lifted through the same proposition context, followed by `rfl`. The generic-first generator prefixes `rfl`, `simp`, and `norm_num` to that bank. Budgets are 2, 4, 8, and 12 actions. Both orders are enumerated, and 200 uniformly randomized orders per pair and configuration use seed `731991`. The primary interval resamples pairs; a second interval clusters the 12 contexts by their eight base templates. Exact order enumeration eliminates Monte Carlo error. The bootstrap intervals measure sensitivity to the constructed pair set.

**Supplementary controls.** The earlier eight-pair protocol gives five order-dependent losses with the pair-specific grammar and the same five with one fixed union grammar over 194 compilations. For the natural case, injecting both histories from their common root and continuing with BFS-Prover-V2 proves every retention arm through another intermediate within 2–4 expansions. The observed natural loss is therefore confined to the aliased node's continuation.

**Step-prover configuration.** Checkpoint `ByteDance-Seed/BFS-Prover-V2-7B`, snapshot `533d59fc6b5e`, with its own tokenizer; prompt is the state's rendering followed by `:::` per the model card; float16 on Apple MPS. Decoding is deterministic beam search, 8 beams, 8 return sequences, at most 64 new tokens, early stopping; the continuation is decoded without special tokens, text after any re-emitted `:::` is kept, and candidates are deduplicated preserving the best sequence score. Tactic execution carries a 60 s watchdog; a watchdog interrupt abandons the class as channel desynchronization. The search queue orders by negated cumulative sequence score with insertion order breaking ties. Per-class first candidate lists, per-search reached-state hashes, and proof traces are in the released run files, together with the three repetition runs and the triple-budget run for the sensitive class.

## E Candidate exposure across policies

**Fixed candidate sets.** For each of the 146 restored members in the original 59-class corpus, we execute the BFS-Prover-V2 top-8 list three

| Policy | $\lvert C\rvert$ | covered | ambiguous | rank 1 | headroom $> 0$ |
|---|---|---|---|---|---|
| ReProver | 59 | 55 | 2 | 0 | 0 |
| miniCTX | 55 | 41 | 2 | 1 | 0 |
| BFS-7B | 59 | 56 | 1 | 0 | 0 |

Table 9: Candidate exposure. "Covered" counts classes with an accepted candidate for every member; "ambiguous" counts classes containing an acceptance-divergent candidate.

times, yielding 3,504 compiler calls and 1,168 state–candidate labels. The lists are identical across members of every alias class, and all repetitions agree.

**Candidate set headroom.** For candidate set $A$ and class $C$, the member-aware oracle averages $\max_{a \in A} r(s_i, a)$ over members. The shared-action oracle maximizes the mean acceptance of one action across members. Their difference measures member-specific headroom within $A$ and has no monotonic relationship with the full tactic language.

For BFS-Prover-V2, the member-aware and shared-action oracles both cover 134/146 members. The corresponding equality holds in every fully covered class for all three generators. These fixed candidate sets provide almost no opportunity for a member-aware score to alter proof availability.

## F Artifact

The supplement is available at `https://github.com/edisonly1/mathnlp1`. It contains the harnesses, per-class outcomes, intervention files, policy candidates, search traces, pinned versions, and seeds. One command reproduces each table from the released records, with live modes for the motivating witness and the current intervention.