LEAN2ISABELLE: VERIFIER-GROUNDED CROSS-ASSISTANT PROOF TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Verified proof artifacts remain tied to the assistant in which they were written. We study Lean-to-Isabelle migration as a verifier-grounded learning problem: given a Lean theorem and proof, construct an Isabelle/HOL theory that passes Isabelle/PISA checking and an audited semantic evaluation against the source theorem. Lean2Isabelle factorizes target-statement alignment from statement-conditioned theory reconstruction and uses Minimal Valid Progress (MVP), a bounded Isabelle proof-context signal, to supplement sparse pass/fail rewards. We construct 26,104 complete-theory Herald-ISA examples and 9,683 LeanWorkbook-ISA statement-level examples, and evaluate external Lean inputs in MiniF2F-DSP (217 examples). On LeanWorkbook-ISA, the full-input cascade reaches 39.1% end-to-end SEMPASS@1, while a statement-only frozen structural translator followed by Isabelle automation reaches 24.3%; with a verified target statement, GRPO-MVP reaches 54.3%, improving over Theory-SFT by 4.5 percentage points. Removing tactics, shuffling proof steps, or retaining only the Lean statement lowers Reference-Statement SEMPASS@1 to 46.9%, 46.1%, and 43.2%, respectively. Together, these results establish a measurable theorem-level interface for cross-assistant proof migration and identify statement alignment, target formulation, and proof construction as distinct bottlenecks.

1 INTRODUCTION

Formal mathematics is distributed across Lean, Isabelle/HOL, Rocq, and other assistants whose foundations, libraries, and proof idioms are not interchangeable. Same-assistant theorem provers can exploit a fixed statement and library context, but cross-assistant reuse must first align the target statement and then reconstruct a kernel-checked target theory. We study this setting as a learning problem rather than as syntax conversion: success requires Isabelle checking and an audited semantic comparison with the source theorem.

The central interface factorizes migration into target-statement prediction and statement-conditioned theory generation. Reference-Statement evaluation supplies a verified Isabelle statement and isolates proof reconstruction; Predicted-Statement evaluation composes statement prediction with theory generation. The factorization exposes whether a failure comes from semantic statement drift, proof reconstruction, or their cascade. It also supports verifier-grounded learning: MVP converts bounded changes in Isabelle proof-context structure into a partial-progress signal while strict Isabelle acceptance remains the correctness criterion.

Cross-assistant migration couples two decisions: formulating a faithful target statement and reconstructing a proof in a different library and proof language. A verified Lean proof can carry theorem-specific structure across this boundary, but only when the target statement preserves the source theorem and the generated Isabelle theory closes in its own context. We make this interface explicit so that each stage can be evaluated independently and its failure mode can be measured.

We build Herald-ISA from complete `FrenzyMath/Herald_proofs` records and retain 26,104 Isabelle-accepted, semantically filtered theories for supervised initialization (Gao et al., 2025). We reconstruct Lean proofs from proved Lean-Workbook traces and retain 9,683 verified statement-level examples for RL and evaluation (Ying et al., 2024; Zheng et al., 2022). MiniF2F-DSP supplies 217 external Lean proof inputs for end-to-end transfer (Ren et al., 2025; Zheng et al., 2022). The training

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.

pipeline uses target-side retrieval only during teacher construction, supervised initialization, and Isabelle/PISA verifier feedback with MVP shaping and a difficulty-ordered curriculum.

Contributions.

- We formulate proof-level cross-assistant migration as a verifier-grounded, factorized learning problem and operationalize MVP as a bounded signal for statement-conditioned Isabelle theory reconstruction.
- We provide verified resources and diagnostics spanning statement alignment, proof reconstruction, source-proof dependence, external-source transfer, and the Reference–Predicted cascade.
- We show that the learned pipeline exceeds a frozen structural translator and that matched Lean proof information contributes beyond the target statement, while quantifying the remaining statement-gate and proof-reconstruction bottlenecks.

2 Related Work

Learning-assisted theorem proving and verifier-guided learning. Learning-assisted provers in Lean and Isabelle combine retrieval, proof search, repair, and verifier feedback, usually within one assistant and its native library (The mathlib Community, 2020; Yang et al., 2023; Xin et al., 2025; Ren et al., 2025; Lin et al., 2026; Jiang et al., 2022; 2023; First et al., 2023; Mikuła et al., 2024; Wang et al., 2026b). GRPO-style and process-oriented objectives use executable feedback for mathematical reasoning (Shao et al., 2024; Guo et al., 2025; Yu et al., 2026; Lightman et al., 2024; Setlur et al., 2025; Cui et al., 2025; Kim & Yun, 2026). We use the same verifier-centered principle at a different interface: the source proof is Lean, the target theory is Isabelle, and correctness also depends on source–target statement preservation.

Faithfulness and formal-library alignment. Formalization benchmarks and semantic-consistency methods show that type-checking does not guarantee preservation of assumptions, domains, library objects, or theorem strength (Zheng et al., 2022; Azerbayev et al., 2023; Lu et al., 2025; Li et al., 2024; Liu et al., 2025; 2026; Cabral et al., 2026; Wang et al., 2026a; Min et al., 2026). Library interchange and concept-alignment efforts study exports, logical frameworks, and mappings between formal libraries (Assaf et al., 2023; Dowek & Thiré, 2023; Kohlhase & Rabe, 2021; Gauthier & Kaliszyk, 2019). Our evaluation combines Isabelle checking with an audited semantic layer to separate formal validity from source–target semantic alignment.

Cross-assistant translation. Multi-system benchmarks and recent translation work span theorem statements, simplified target languages, and Lean–Rocq or Lean–Dafny artifacts (Zheng et al., 2022; Tsoukalas et al., 2024; Xu et al., 2026a; Li et al., 2025; Viennot et al., 2025; Baksys et al., 2025; Stoskopf et al., 2025; Xu et al., 2026b). Lean2Isabelle extends this line to complete Lean-proof to Isabelle-theory migration, combining target verification, source-aware ablations, and external-source transfer in a factorized statement/proof interface.

3 Method

3.1 Overview

Given a Lean theorem and proof (de Moura & Ullrich, 2021; The mathlib Community, 2020), verified Lean-to-Isabelle translation must produce a complete Isabelle/HOL theory (Nipkow et al., 2002) accepted through PISA (Jiang et al., 2021). Let $x = (s_L, p_L)$ denote the Lean statement/proof, $z = (I, C, t)$ an Isabelle target specification with imports I, optional local context C, and theorem statement t, and y a complete Isabelle theory. The proof model must prove the supplied Isabelle statement rather than rewrite it, and success requires both Isabelle checking and LLM-based semantic evaluation.

Lean2Isabelle factorizes this task into target-statement prediction and statement-conditioned theory generation. We train a statement translator $q_\phi(z \mid s_L)$ and a proof model $\pi_\theta(y \mid x, z)$; Reference-Statement translation conditions on a verified $z^\star$, while Predicted-Statement translation composes

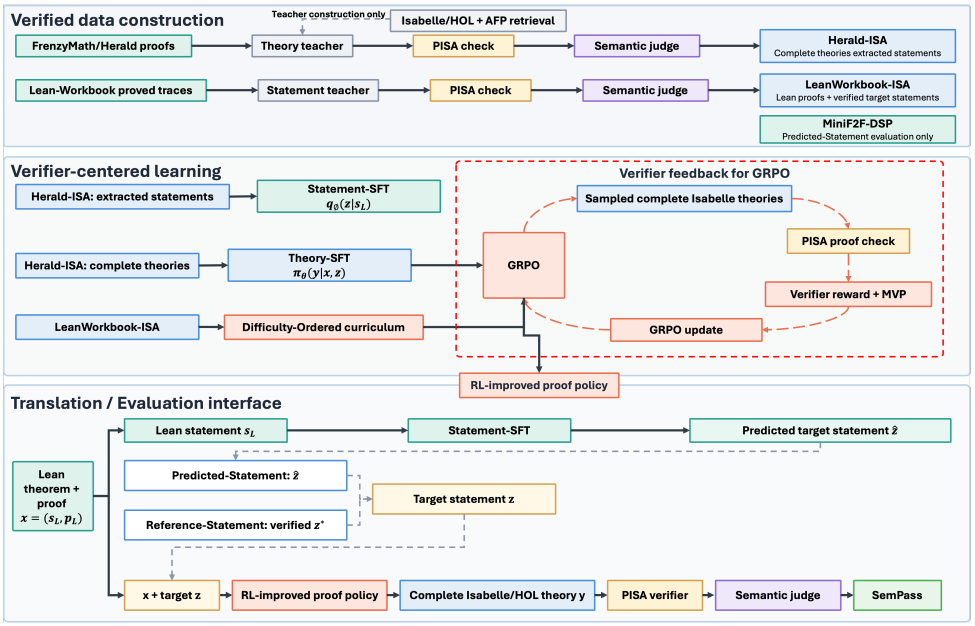


Figure 1: Overview of the verified Lean-to-Isabelle framework: verified data construction, statement/proof models, Isabelle/PISA verifier feedback, and Reference-Statement versus Predicted-Statement evaluation.

$\hat{z} \sim q_\phi(\cdot \mid s_L)$ with $y \sim \pi_\theta(\cdot \mid x, \hat{z})$. This factorization exposes statement drift separately from proof reconstruction.

Figure 1 summarizes the framework. We first construct verified teacher targets, then train retrieval-free Statement-SFT and Theory-SFT models, and finally optimize the proof model with Isabelle verifier feedback, Minimal Valid Progress (MVP), and a difficulty-ordered curriculum.

3.2 Verified Translation Data Construction

Cross-system proof translation requires paired data whose target side is executable and whose source–target relation is explicitly evaluated. We retain statement targets only when they check inside a minimal Isabelle theory and pass statement-level semantic evaluation; complete theories must additionally pass Isabelle/PISA and proof-level semantic evaluation. The semantic gates test domains, assumptions, quantifiers, goals, library objects, and theorem strength. The resulting resources support controlled analysis of statement alignment, proof reconstruction, and cascade failures; Appendix A specifies the filters, and Section 4.1 describes the audit protocol.

Herald-ISA provides complete-theory supervision. The Theory Teacher translates complete `FrenzyMath/Herald_proofs` Lean records (Gao et al., 2025) into Isabelle theories, optionally using the retrieval module in Section 3.3; candidate theories are checked by Isabelle/PISA, revised from verifier diagnostics when needed, and then filtered by proof-level semantic evaluation. We extract the target statement from each accepted theory, yielding $N_{\mathrm{H}} = 26{,}104$ verified examples: the 25,581-example training split initializes Theory-SFT and Statement-SFT, while the 261/262 validation/test splits are kept out for evaluation.

LeanWorkbook-ISA provides the main RL and evaluation inputs. From proved `internlm/Lean-Workbook` tactic-state traces, we reconstruct complete Lean proofs by grouping proved theorem traces and concatenating tactics; the Statement Teacher then translates each Lean statement into a minimal Isabelle target specification, revising invalid or semantically rejected candidates with verifier and semantic feedback. After miniF2F-test overlap removal (Zheng et al., 2022), Isabelle checking, and statement-level semantic evaluation, the final resource contains $N_{\mathrm{LW}} = 9{,}683$ examples with 9,198/242/243 train/validation/test splits. Unlike Herald-ISA, LeanWorkbook-ISA stores reconstructed Lean proofs and verified Isabelle statements but no reference Isabelle proofs, so RL receives the source proof and $z_i^\star$ but must synthesize the Isabelle

theory. We also derive MiniF2F-DSP from public DeepSeek-Prover-V2 miniF2F Lean proofs for Predicted-Statement evaluation (Ren et al., 2025; Zheng et al., 2022).

3.3 Retrieval-assisted theory teacher

Retrieval is used only while constructing Theory-Teacher targets: an Isabelle/HOL/AFP index supplies candidate identifiers, imports, and code, after which Isabelle/PISA and semantic evaluation determine correctness. Statement Teacher and all student models are retrieval-free at evaluation time. Appendix F gives the index and prompt details; Appendix L.5 tests whether injecting the same top-20 evidence at student inference resolves the proof bottleneck.

3.4 Supervised initialization

Supervised initialization gives the student models a reliable Isabelle/HOL output distribution before sparse verifier-based RL (Shao et al., 2024; Yu et al., 2026; Ren et al., 2025; Wang et al., 2026c). We therefore train both SFT stages on Herald-ISA demonstrations with the standard target-token-masked conditional language-modeling loss. Statement-SFT maps Lean imports and a Lean formal statement to a verified Isabelle target specification; it predicts target statements for the Predicted-Statement setting and is not trained to construct proofs. Theory-SFT maps Lean imports, a Lean formal proof, and a fixed verified Isabelle statement to a complete Isabelle theory proving that statement without unfinished proofs; it initializes RL. Because both stages are derived from the same filtered Herald-ISA examples, the predicted statement format remains aligned with the proof model's conditioning interface.

All student models use the same DeepSeek-Prover-V2-7B pretrained weights as the backbone (Ren et al., 2025). Statement-SFT and Theory-SFT are full supervised fine-tunes, while the GRPO stage initializes from Theory-SFT and updates a LoRA adapter; Appendix E reports the optimization settings.

3.5 Verifier-based RL and Minimal Valid Progress

Verifier-based RL further optimizes the Theory-SFT policy using target proof-assistant execution (Ren et al., 2025). Given a Lean proof and Isabelle theorem statement, the policy samples complete Isabelle theories; admissible samples are checked by Isabelle/PISA for pass/fail feedback, diagnostics, and proof-level traces.

Strict Isabelle pass/fail feedback is the authoritative correctness signal. For input q, we sample G completions $\{c_i\}_{i=1}^{G}$. Passing completions receive the strict verifier reward; failed completions receive zero verifier reward by default; structurally invalid or unfinished outputs receive a small admissibility penalty. This signal is reliable but sparse.

Minimal Valid Progress (MVP) provides a bounded partial-progress score for failed completions with proof-level traces. Let ℓ_r^{before} and ℓ_r^{after} denote the proof-context levels before and after the r-th proof-relevant execution step. We define

$$O(c) = \sum_r \max(0, \ell_r^{\text{after}} - \ell_r^{\text{before}}), \qquad C(c) = \sum_r \max(0, \ell_r^{\text{before}} - \ell_r^{\text{after}}),$$
$$\text{MVP}(c) = \frac{\min(O(c), C(c))}{\max(1, O(c))}. \tag{1}$$

MVP rewards closing opened local proof contexts rather than being longer, executing further, or failing later: unclosed proof blocks receive little credit, while failures after some closures receive partial progress.

The scalar reward combines strict verification with MVP shaping:

$$R(c_i) = R_{\text{adm}}(c_i) + \mathbf{1}[\text{pass}(c_i)] + \mathbf{1}[\text{adm}(c_i) \wedge \text{fail}(c_i) \wedge \text{trace}(c_i)]\lambda\,\text{MVP}(c_i), \tag{2}$$

where $R_{\text{adm}} \leq 0$ denotes admissibility penalties, $\text{adm}(c_i)$ means structurally admissible, and $\lambda < 1$ bounds failed-completion shaping below strict success. Since $0 \leq \text{MVP}(c) \leq 1$, a verifier-passing proof always outranks a failed proof; invalid outputs receive no positive MVP shaping. For each input group, we compute the normalized advantage $A_i = \frac{R(c_i)-\mu_q}{\sigma_q+\epsilon}$, where μ_q and σ_q are the mean and standard deviation of rewards within the G completions for q. With token mask $m_{i,t}$ and importance ratio $\rho_{i,t} = \frac{\pi_\theta(c_{i,t}|q,c_{i,<t})}{\pi_{\text{old}}(c_{i,t}|q,c_{i,<t})}$, we minimize a GRPO-style group-relative objective with DAPO-inspired token-level normalization and asymmetric clipping

$$\mathcal{L}(\theta) = -\frac{1}{\sum_{i,t} m_{i,t}} \sum_{i,t} m_{i,t} \min(\rho_{i,t}A_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})A_i) + \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \quad (3)$$

The objective combines GRPO-style group-relative advantages with DAPO-inspired token aggregation, length normalization, and asymmetric clipping (Yu et al., 2026). Unlike dense process-supervision methods (Lightman et al., 2024; Setlur et al., 2025; Cui et al., 2025; Kim & Yun, 2026), MVP is a bounded PISA-trace proxy that ranks failed completions without relaxing the formal success criterion. The resulting policy keeps the same statement-conditioned interface as Theory-SFT.

3.6 Difficulty-Ordered curriculum

The Difficulty-Ordered curriculum presents RL prompts from easier, verifier-informative examples toward structurally harder ones. Early RL is inefficient when many inputs yield no verifier-passing samples and when frequent import changes increase Isabelle checking cost. We therefore compute a deterministic ordering prior before RL from teacher-observed alignment effort, source proof structure, and Isabelle import signatures; this is conceptually related to GAR's difficulty-aligned implicit curriculum (Wang et al., 2026c). The prior first groups examples into coarse stages using the Statement Teacher effort needed to obtain a verified semantic match and the structural size of the Lean proof/theorem, then uses proof-form tags and import-signature grouping as within-stage tie-breakers. The curriculum affects only input order: rewards still come from Isabelle/PISA checking and MVP shaping, and no curriculum signal is used as a success criterion. Appendix D.1 describes the frozen signal families and ordering contract.

4 Experiments

4.1 Experimental setup

Our experiments test three diagnostic claims for Lean-to-Isabelle proof translation: statement drift is measurable, proof reconstruction improves with verifier-guided RL, and Lean source inputs provide signal beyond the target Isabelle statement. The evaluation is organized accordingly. Statement Translation tests the first-stage target specification. Reference-Statement Proof Translation gives the proof model a verified Isabelle statement and isolates proof reconstruction. Predicted-Statement End-to-End Translation uses the Statement-SFT prediction and measures the full cascade. LeanWorkbook-ISA is the primary RL and evaluation domain, Herald-ISA supplies SFT supervision and Reference-Statement evaluation, and MiniF2F-DSP tests transfer to miniF2F-derived proof inputs; Appendix A reports sources, roles, splits, and filtering.

We compare target-only Isabelle automation, a frozen structural translator, supervised theory reconstruction, and verifier-guided training under a common PISA backend. These conditions isolate target-side solvability, learned cross-system alignment, source-proof conditioning, and verifier-guided optimization.

We report Pass@K, EqStmt@1, and SemPass@K. EqStmt@1 adds statement-level semantic evaluation, while SemPass@K requires Isabelle verification and proof-level semantic acceptance. In Predicted-Statement evaluation, the predicted statement must also pass the statement gate. SemPass is an audited operational metric rather than a machine-checked source-preservation certificate. Judge-accepted candidates are manually labeled without consulting the automatic judge rationales: manual acceptance is 94.7% (Reference, $n = 150$) and 98.8% (Predicted, $n = 80$). The full accepted/rejected-bucket audit and post-stratified sensitivity analysis appear in Appendix B.

Table 1: Statement translation with Statement-SFT. Metrics are percentages; N is the number of evaluated examples.

Dataset	Method	PASS@1↑ (%)	EQSTMT@1↑ (%)	N
LeanWorkbook-ISA	Statement-SFT	88.9	86.0	243
Herald-ISA	Statement-SFT	67.6	57.6	262
MiniF2F-DSP	Statement-SFT	88.0	85.7	217

4.2 STATEMENT TRANSLATION

The first diagnostic claim is that the task can detect statement drift rather than merely compile target statements. Statement-SFT reaches 86.0% and 85.7% EQSTMT@1 on LeanWorkbook-ISA and MiniF2F-DSP, while Herald-ISA is harder because broader namespaces and target-side scaffolding make statement translation less uniform. The gap between PASS@1 and EQSTMT@1 shows that an Isabelle-checking statement can still exhibit type or domain shifts, assumption or quantifier changes, operator/library mismatches, or trivialized targets; Appendix J.4 gives a verifier-passing but semantically drifted example.

4.3 STRUCTURAL TRANSLATION BASELINE

To separate learned cross-system alignment from local syntax rewriting, we froze a parameter-free translator that reads only the Lean statement, applies parser and symbol rules, imports `Complex_Main`, and tries a fixed Isabelle method portfolio. Every parser, translation, statement, or proof failure remains in the full denominator. The baseline reaches 24.3% end-to-end SEMPASS@1 on LeanWorkbook-ISA, compared with 39.1% for the learned cascade. On the more library-dependent Herald-ISA split, its parser and translation coverage fall to 28.2% and 1.9%, and its end-to-end score is 0.8%; the learned Statement-SFT reaches 67.6%/57.6% PASS@1/EQSTMT@1. This baseline isolates the contribution of learned alignment beyond frozen local rewriting; Appendix L.1 reports its coverage and failure profile.

4.4 REFERENCE-STATEMENT PROOF TRANSLATION

The second diagnostic claim is that Lean2Isabelle improves proof reconstruction once statement translation is removed from the loop. In Reference-Statement evaluation, the target Isabelle statement is supplied, so the comparison isolates theory reconstruction rather than statement prediction. Target-only automation measures target-side solvability, Theory-SFT measures supervised reconstruction, and GRPO tests whether verifier-guided learning yields further gains.

The Reference-Statement block of Table 2 makes two comparisons. First, task-specific Isabelle theory training accounts for most of the gain over target-only automation. Second, GRPO-MVP-DifficultyOrdered adds consistent improvements over Theory-SFT on both LeanWorkbook-ISA and Herald-ISA. This supports the claim that verifier-guided learning improves proof reconstruction after statement translation is removed from the loop.

4.5 EQUAL-BUDGET RL ABLATION

The RL ablation tests whether the gain comes from verifier-structured learning rather than extra generation budget. Under the same 300-step LeanWorkbook-ISA budget, Theory-SFT, verifier-only Difficulty-Ordered GRPO, and GRPO-MVP-DifficultyOrdered obtain 49.8%, 51.0%, and 52.3% SEMPASS@1, respectively. The full table and training diagnostics are in Appendix L.3; the final GRPO-MVP-DifficultyOrdered model uses the same recipe with a validation-selected longer checkpoint (Appendix D).

4.6 PREDICTED-STATEMENT END-TO-END TRANSLATION

Predicted-Statement evaluation measures the full translation cascade with predicted target Isabelle statements. The lower block of Table 2 separates two effects: GRPO still improves the proof model after statement prediction, but the cascade loses success when the statement stage drifts.

Table 2: Proof translation under Reference-Statement and Predicted-Statement evaluation. Metrics are percentages; N is the number of evaluated examples. The notation $\mu \pm \sigma$ indicates the mean and the standard deviation. Reference-Statement isolates proof reconstruction with a verified Isabelle statement; Predicted-Statement measures the full cascade with the Statement-SFT prediction. Target-only automation is deterministic, so @4 entries are N/A.

Dataset	Condition / proof model	PASS@1↑ (%)	SEMPASS@1↑ (%)	PASS@4↑ (%)	SEMPASS@4↑ (%)	N
Reference-Statement: verified target statement, proof reconstruction isolated						
LeanWorkbook-ISA	Isabelle automation (target only)	37.4	37.4	N/A	N/A	243
LeanWorkbook-ISA	Theory-SFT	50.6	49.8	56.1±0.9	55.4±0.8	243
LeanWorkbook-ISA	GRPO-MVP-DifficultyOrdered	54.7	54.3	60.1±0.4	59.5±0.6	243
Herald-ISA	Isabelle automation (target only)	50.4	50.4	N/A	N/A	262
Herald-ISA	Theory-SFT	63.4	59.9	73.0±1.2	67.5±1.2	262
Herald-ISA	GRPO-MVP-DifficultyOrdered	67.6	64.5	76.8±1.5	71.1±1.3	262
Predicted-Statement: predicted target statement, full cascade						
LeanWorkbook-ISA	Theory-SFT	39.5	37.0	46.1±1.1	44.3±1.0	243
LeanWorkbook-ISA	GRPO-MVP-DifficultyOrdered	41.6	39.1	49.5±0.6	47.5±0.9	243
MiniF2F-DSP	Theory-SFT	21.2	20.7	26.3±1.3	25.4±1.2	217
MiniF2F-DSP	GRPO-MVP-DifficultyOrdered	24.0	24.0	30.3±1.0	29.2±0.7	217

On LeanWorkbook-ISA, GRPO-MVP-DifficultyOrdered reaches 39.1% SEMPASS@1 and 47.5% SEMPASS@4, improving over Theory-SFT by +2.1 and +3.2 percentage points but remaining 15.2 points below the same proof model with reference statements. On MiniF2F-DSP, the final system reaches 24.0% SEMPASS@1 and 29.2% SEMPASS@4, improving over Theory-SFT by +3.3 and +3.8 points. MiniF2F-DSP provides an external transfer setting built from miniF2F-derived Lean proof inputs.

ID-aligned pairing attributes 22 of the 37 lost successes (59.5%) to statement-gate failures and 15 (40.5%) to proof reconstruction under alternative but gate-passing target formulations. On the external MiniF2F-DSP inputs, which are withheld from training and have no teacher Isabelle target, the system reaches 85.7% statement EQSTMT@1 and 24.0% end-to-end SEMPASS@1; the full pairing matrix and transfer protocol are in Appendix L.6.

The failure taxonomy gives the same decomposition at the example level. On LeanWorkbook-ISA, 34 of 243 predicted statements fail the statement semantic gate, 108 remain proof failures after statement evaluation, and only 6 verifier-passing theories are rejected by the proof-level semantic gate. On MiniF2F-DSP, 31 examples fail statement evaluation and 134 fail during proof construction after the statement gate. The dominant loss therefore occurs before or during target-theory reconstruction, rather than in the final semantic audit.

The agreement between the paired and example-level counts shows that the aggregate Reference–Predicted gap is not driven by a small set of anomalous cases. It reflects two distinct operations: target-statement alignment and target-theory reconstruction. The source-proof ablations below hold the target statement fixed and therefore measure the information available to the proof model after alignment succeeds. This stage-aware reading lets each training change be evaluated against the capability it is intended to improve. This separation is also why the proof-input ablations use a fixed target: they test source-proof use after statement alignment rather than mixing the two stages.

4.7 DOES THE MODEL USE THE LEAN PROOF?

The third diagnostic claim is that the translator uses Lean-side information beyond the reference Isabelle statement. With the proof model and reference Isabelle statement fixed, removing all Lean source input lowers GRPO-MVP-DifficultyOrdered SEMPASS@1 from 54.3% to 45.3%, and replacing the matched Lean proof with an unrelated proof lowers it to 43.2%; Theory-SFT shows the same pattern (Table 3).

Equal-budget controls show that the final gain is incremental but reproducible: under 300 RL steps, Theory-SFT, verifier-only Difficulty-Ordered GRPO, and MVP with Difficulty-Ordered inputs obtain 49.8%, 51.0%, and 52.3% SEMPASS@1, respectively. The full table and training diagnostics are in Appendix L.3; the selected final checkpoint uses the same recipe with validation-based stopping. Because the checkpoint and target statement stay fixed, this comparison attributes the change to source-input content and verifier-guided optimization rather than target formulation.

Table 3: Lean proof input ablation on LeanWorkbook-ISA ($N = 243$). Full input denotes the Lean statement, matched Lean proof, and reference Isabelle statement; Table 23 gives the complete expanded transform table.

Input condition	GRPO SEMPASS@1
Full input	54.3
Tactics removed	46.9
Within-proof step shuffle	46.1
Reference statement only	45.3
Proof-skeleton-only	43.6
Lean statement-only	43.2
Cross-sample shuffled proof	43.2

These controls use the same checkpoint, reference statement, decoding, and evaluation pipeline; only the visible Lean input changes. Removing tactics or shuffling proof units lowers SemPass by 7.4 or 8.2 points, and neither a generic proof skeleton nor the Lean statement alone exceeds the reference-statement-only control. The matched Lean proof therefore provides theorem-specific signal beyond the target statement; full protocols and Pass@4 results appear in Appendix L.2.

4.8 RESOURCE SCOPE AND TARGET-CONTEXT COVERAGE

The two paired resources expose complementary parts of the migration problem. LeanWorkbook-ISA is concentrated around compact theorems: 72.43% of its examples are elementary arithmetic/algebra, 41.40% are inequality/order statements, and 40.39% use arithmetic or algebraic automation. Its targets use 38 unique imports, with 4.1% requiring a non-base import and 0.5% introducing auxiliary scaffolding. Herald-ISA is broader and more context-dependent: its largest domain groups are analysis/topology/measure (15.29%), abstract algebra (13.24%), and logic/sets/order (12.93%), while rewriting (58.91%), structured reasoning (19.91%), induction (11.05%), and extensionality (7.20%) cover its main proof styles. Its targets use 829 unique imports; 34.3% require non-base imports and 67.4% introduce definitions, locales, records, classes, or helper lemmas before the final theorem.

These distributions explain why the frozen local-rewrite baseline degrades sharply on Herald-ISA. A source-only audit finds small coarse composition shifts after filtering (0.12/0.06 percentage points in LeanWorkbook-ISA and 1.13/0.90 in Herald-ISA for domain/proof style). Together, the paired resources isolate statement alignment and proof reconstruction, while MiniF2F-DSP adds an external-source transfer test without a reference Isabelle target.

Taken together, the diagnostics identify a consistent hierarchy of bottlenecks. In the 37-ID Reference–Predicted gap decomposition, 22 net losses (59.5%) occur at the statement gate and 15 (40.5%) remain after the target statement is verified. When that gate is supplied, top-20 test-time retrieval decreases SemPass from 54.3% to 49.0%, while sequential Repair-B4 adds only about 1.2 points. These controls separate target formulation from proof construction and show why migration requires more than theorem-string translation.

4.9 VALIDATION OF THE MIGRATION INTERFACE

The evaluation combines two checks that answer different questions about a translated artifact. Isabelle/PISA establishes that the target theory is executable and kernel-accepted. The semantic gate then checks whether the target statement and proof preserve the source theorem's domain, assumptions, quantifiers, mathematical objects, and conclusion. In a manual audit of verifier-passing candidates, the automatic proof-level gate had 94.7% manual acceptance on Reference-Statement candidates ($n = 150$) and 98.8% on Predicted-Statement candidates ($n = 80$); manual labels were assigned from the Lean input, supplied Isabelle statement, and generated theory.

The paired resources also retain source-side breadth after construction filtering. Mean absolute shifts between raw and retained cohorts are 0.12/0.06 percentage points for the LeanWorkbook-ISA domain and proof-style distributions and 1.13/0.90 points for Herald-ISA. The audit therefore supports using the resources to study alignment and reconstruction across distinct source distributions, while target-context selection remains part of the measured translation setting. MiniF2F-DSP adds a separate external-source transfer test: its 217 Lean-verified inputs have no reference Isabelle target at inference and reach 85.7% statement EQSTMT@1 and 24.0% end-to-end SEMPASS@1.

Together, these checks define the evidence boundary for the main task. The verifier provides a deterministic target-validity test, semantic evaluation measures source–target correspondence, and the external split tests whether the factorized interface extends beyond the resources used for supervised initialization and RL. The resulting measurements support the central claim at the level of theorem translation and expose the two stages that future improvements must address: target formulation and target-theory reconstruction.

4.10 Implications for cross-assistant evaluation

The factorization also changes how cross-assistant systems can be compared. A target statement that passes Isabelle/PISA but fails the semantic gate indicates a translation error, whereas a semantically aligned statement whose theory fails indicates a reconstruction error. Reporting both values therefore separates theorem preservation from target-language proof search. In the paired Reference–Predicted comparison, 59.5% of the net loss occurs before the proof gate, making target formulation the first optimization target rather than an incidental preprocessing step.

Once a target statement is fixed, the source proof remains useful information for target-theory construction. Removing tactics or shuffling proof units lowers SemPass by 7.4 or 8.2 points, while a generic proof skeleton and the Lean statement alone do not exceed the reference-statement-only control. This pattern supports a staged design: first align the mathematical interface, then use theorem-specific proof structure to guide reconstruction in the target context.

The external MiniF2F-DSP split tests the same interface without a reference Isabelle target at inference. Its 85.7% statement EqStmt@1 and 24.0% end-to-end SemPass@1 show that the decomposition remains measurable on a new source distribution, while the gap between the two metrics identifies target-theory construction as the next system-level bottleneck.

5 Limitations

The paired resources are constructed through teacher translation, Isabelle verification, and semantic filtering and are used to study theorem-level migration. A source-only audit finds little coarse distribution shift in LeanWorkbook-ISA and modest reweighting in Herald-ISA, while within-category target-side selection remains possible (Appendix L.7). SemPass combines Isabelle verification with audited semantic evaluation; the judge audit reports accepted-bucket precision and conservative rejections (Appendix B). With a verified target statement, proof construction remains the main empirical bottleneck: top-20 test-time retrieval lowers SemPass from 54.3% to 49.0% in the frozen setup, while sequential Repair-B4 adds about 1.2 points (Appendix L.5).

The central empirical implication is that cross-assistant migration should be optimized as a semantic interface, rather than as target-language generation alone. Statement evaluation measures whether the source theorem has been carried into the target library with its mathematical content intact; Reference-Statement evaluation then measures whether the target theory can be reconstructed once that interface is fixed. This separation turns a single translation score into two actionable capabilities and connects each failure to a concrete stage of artifact migration.

6 Conclusion

Lean2Isabelle defines a factorized, verifier-grounded interface for theorem-level Lean-to-Isabelle proof migration. Statement evaluation exposes semantic drift, verifier-guided training improves statement-conditioned theory reconstruction, and source-input controls show that matched Lean proofs contribute information beyond the target statement. The learned cascade outperforms frozen structural translation, while the Reference–Predicted comparison identifies target formulation and proof construction as the dominant remaining bottlenecks. This decomposition provides a concrete basis for improving cross-assistant proof-artifact migration.

The practical consequence is an evaluation protocol in which every candidate yields a verifier result, a semantic result, and a stage-specific failure label. Improvements can therefore be attributed to statement alignment, target-context reconstruction, or source-proof conditioning instead of being absorbed into one end-to-end score. The external split shows that this protocol remains informative without a reference Isabelle target, making the interface useful both as a translation task and as a testbed for source-conditioned formal reasoning across proof assistants.

Paired and external splits provide controlled and transfer operating points, making each verified theorem artifact a common unit for cross-assistant reasoning.

AI USE DISCLOSURE

Large language models were used to assist with synthetic formal-data generation and filtering, experimental code and analysis, and manuscript editing. All reported formal results were checked with the stated Isabelle/PISA or Lean verifiers, and the authors remain responsible for the final experiments, claims, and text.

REFERENCES

Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti: a logical framework based on the $\lambda\Pi$-calculus modulo theory, 2023. URL https://arxiv.org/abs/2311.07185.

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

Mantas Baksys, Stefan Zetzsche, Olivier Bouissou, and Sean B. Holden. MiniF2F-Dafny: LLM-guided mathematical theorem proving via auto-active verification, 2025. URL https://arxiv.org/abs/2512.10187.

Rafael Medeiros Cabral, Tuan Manh Do, Xuejun Yu, Wai Ming Tai, Zijin Feng, and Xin Shen. ProofFlow: A dependency graph approach to faithful proof autoformalization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=s9t2FJVsBH.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, volume 12699 of *Lecture Notes in Computer Science*, pp. 625–635. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

Gilles Dowek and François Thiré. Logipedia: a multi-system encyclopedia of formal proofs, 2023. URL https://arxiv.org/abs/2305.00064.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1229–1241. Association for Computing Machinery, 2023. doi: 10.1145/3611643.3616243.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated Lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Se6MgCtRhz.

Thibault Gauthier and Cezary Kaliszyk. Aligning concepts across proof assistant libraries. *Journal of Symbolic Computation*, 90:89–123, 2019. doi: 10.1016/j.jsc.2018.04.005.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Albert Qiaochu Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. LISA: Language models of ISAbelle proofs. In *Proceedings of the 6th Conference on Artificial Intelligence and Theorem Proving*, pp. 63–65, 2021. URL https://aitp-conference.org/2021/abstract/paper_17.pdf.

Albert Qiaochu Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 8360–8373, 2022.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SMa9EAovKMC.

Minsu Kim and Se-Young Yun. Process-verified reinforcement learning for theorem proving via Lean. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=P00k4DFaXF.

Michael Kohlhase and Florian Rabe. Experiences from exporting major proof assistant libraries. *Journal of Automated Reasoning*, 65:1265–1298, 2021. doi: 10.1007/s10817-021-09604-0.

Xinrui Li, Wenjie Ma, Hangrui Bi, Zhaoyu Li, Xujie Si, and Kaiyu Yang. ProofGym: Unifying LLM-based theorem proving across formal systems. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=RrSQxcg6Nu. Workshop paper.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Fan Yang, Xian Zhang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *Advances in Neural Information Processing Systems*, volume 37, pp. 53598–53625, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=j4C0nALrgK.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hUb2At2DsQ.

Xiaoyang Liu, Tao Zhu, Zineng Dong, Yuntian Liu, Qingfeng Guo, Zhaoxuan Liu, Yu Chen, and Tao Luo. ASSESS: A semantic and structural evaluation framework for statement similarity. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=avwNGWtiHF.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. FormalAlign: Automated alignment evaluation for autoformalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=B5RrIFMqbe.

Maciej Mikuła, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Albert Q. Jiang, Jin Peng Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oYjPk8mqAV.

Marcus J. Min, Yeqi Gao, Wilson Sy, Zhaoyu Li, Xujie Si, and Osbert Bastani. Divide and abstract: Autoformalization via decomposition and abstraction learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=NjgaeXNit3.

Tobias Nipkow, Markus Wenzel, and Lawrence C. Paulson (eds.). *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. doi: 10.1007/3-540-45949-9.

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, 2025. URL https://arxiv.org/abs/2504.21801.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=A6Y7AqlzLW.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Théo Stoskopf, Cyril Cohen, and Nicolas Tabareau. Babel-formal: Translation of proofs between Lean and Rocq. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=WBI1PL0hZ2. Workshop paper.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 367–381. Association for Computing Machinery, 2020. doi: 10.1145/3372885.3373824.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11545–11569, 2024.

Jules Viennot, Guillaume Baudart, Emilio Jesús Gallego Arias, and Marc Lelarge. Minif2f in rocq: Automatic translation between proof assistants — a case study. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=wkELXtGZa6. Workshop paper.

Hanyu Wang, Ruohan Xie, Yutong Wang, Guoxiong Gao, XintaoYu, and Bin Dong. Aria: an agent for retrieval and iterative auto-formalization via dependency graph. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=CPxZClPMiy.

Jianing Wang, Jianfei Zhang, Qi Guo, Linsen Guo, Rumei Li, Chao Zhang, Chong Peng, Cunguang Wang, Dengchang Zhao, Jiarong Shi, Jingang Wang, Liulin Feng, Mengxia Shen, Qi Li, Shengnan An, Shun Wang, Wei Shi, Xiangyu Xi, Xiaoyu Li, Xuezhi Cao, Yi Lu, Yunke Zhao, Zhengyu Chen, Zhimin Lin, Wei Wang, Peng Pei, and Xunliang Cai. LongCat-Flash-Prover: Advancing native formal reasoning via agentic tool-integrated reinforcement learning, 2026b. URL https://arxiv.org/abs/2603.21065.

Ruida Wang, Jiarui Yao, Rui Pan, Shizhe Diao, and Tong Zhang. GAR: Generative adversarial reinforcement learning for formal theorem proving. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=1MUZsrJxi9.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Haowei Zhang, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4YAIwrsXa.

Qiyuan Xu, Xiaokun Luan, Renxi Wang, Joshua Ong Jun Leang, Peixin Wang, Haonan Li, Wenda Li, and Conrad Watt. Neural theorem proving for verification conditions: A real-world benchmark. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=MfDyickxQA.

Qiyuan Xu, Renxi Wang, Peixin Wang, Haonan Li, and Conrad Watt. A minimalist proof language for neural theorem proving over Isabelle/HOL. *Proceedings of the ACM on Programming Languages*, 10(OOPSLA1): 2073–2098, 2026b. doi: 10.1145/3798275.

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*, volume 36, pp. 21573–21612, 2023.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems*, volume 37, pp. 105848–105863, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A Dataset Construction and Filtering

Table 4 summarizes the data domains used in the main experiments, and Table 5 summarizes the retained artifacts and split sizes. Herald-ISA supplies complete Isabelle theories for supervised initialization. LeanWorkbook-ISA supplies reconstructed Lean proof inputs and verified Isabelle statements for RL and main evaluation. MiniF2F-DSP supplies miniF2F-derived Lean proof inputs for Predicted-Statement evaluation; it is a proof-translation subset because the end-to-end pipeline requires Lean proof inputs, whereas the official miniF2F benchmark does not provide them.

The formal filters used in data construction mirror the main evaluation metrics. Let $\mathcal{V}_{\mathrm{stmt}}(z)$ denote Isabelle/PISA checking of a target specification inside a minimal Isabelle theory, and let $\mathcal{V}(y)$ denote Isabelle/PISA checking of a complete Isabelle theory. Let $\mathcal{J}_{\mathrm{stmt}}$ and $\mathcal{J}_{\mathrm{pf}}$ denote the statement- and proof-level semantic-evaluation judges. We retain a teacher candidate only when

$$\mathrm{Accept}_{\mathrm{stmt}}(s_L, z) = \mathcal{V}_{\mathrm{stmt}}(z) \cdot \mathcal{J}_{\mathrm{stmt}}(s_L, z) = 1, \tag{4}$$

$$\mathrm{Accept}_{\mathrm{theory}}(x, z, y) = \mathcal{V}(y) \cdot \mathcal{J}_{\mathrm{pf}}(x, z, y) = 1. \tag{5}$$

Here $x = (s_L, p_L)$ is the Lean statement/proof input, z is the Isabelle target specification, and y is the complete Isabelle theory.

Table 4: Data domains used in the main experiments.

Domain	Source	Main role	Eval N
Herald-ISA	`FrenzyMath/Herald_proofs`	SFT supervision; Reference-Statement evaluation	262
LeanWorkbook-ISA	Lean-Workbook proved tactic traces	RL training; Reference- and Predicted-Statement evaluation	243
MiniF2F-DSP	DeepSeek-Prover-V2 miniF2F Lean proofs	Predicted-Statement evaluation on miniF2F-derived inputs	217

Table 5: Dataset construction and filtering summary. Raw, Kept, Train, Val, and Test are counts. N/A indicates that no train/validation split is defined for the evaluation-only resource.

Artifact	Raw	Kept	Train	Val	Test	Filtering criteria
Herald-ISA	44,553	26,104	25,581	261	262	Isabelle checking through PISA, proof-level semantic evaluation
LeanWorkbook-ISA	10,268	9,683	9,198	242	243	proved trace reconstruction, statement translation, miniF2F overlap removal, semantic statement acceptance
MiniF2F-DSP	217	217	N/A	N/A	217	DeepSeek-Prover-V2 Lean-verified subset extraction and LeanWorkbook overlap check

Table 6 reports the overlap checks used for the final LeanWorkbook-ISA and MiniF2F-DSP splits. These checks cover exact and normalized duplicates across the evaluation resources.

Table 6: Overlap and duplicate checks for the evaluation resources.

Check	Scope	Result	Notes
miniF2F-test overlap removal	LeanWorkbook-ISA	6 removed	exact/normalized source matching
duplicate theorem IDs	LeanWorkbook-ISA train/val/test	0 found	split-level ID check
normalized Lean formal / NL duplicates	LeanWorkbook-ISA splits	0 / 0 found	exact normalized source-side matching
MiniF2F-DSP vs. LeanWorkbook-ISA	cross-dataset	0 formal / 0 NL found	exact normalized Lean formal and natural-language checks

B SEMANTIC JUDGE PROTOCOL AND EXAMPLE

The main text reports the aggregate manual audit of the automatic proof-level semantic judge in Table 7. This appendix records the audit protocol and one representative positive case. The judge is not a student checkpoint, and it is used only for semantic-evaluation filtering and evaluation, not for generating the proof candidates being evaluated. In Predicted-Statement evaluation, candidates must first pass the statement evaluation; the proof-level judge then checks theorem similarity and proof acceptability for verifier-passing theories under the supplied Isabelle specification. Manual labels are assigned by inspecting the Lean input, the supplied Isabelle specification, and the verifier-passing Isabelle theory; the automatic judge rationales are not consulted when assigning manual labels. The audit samples 300 verifier-passing candidates, stratified by evaluation stage: 150 Reference-Statement and 80 Predicted-Statement judge-accepted candidates estimate false-positive risk, while 40 Reference-Statement and 30 Predicted-Statement judge-nonaccepted candidates diagnose conservative rejections. FPR is computed within the judge-accepted bucket, and the upper-bound FPR treats uncertain accepted-bucket cases as incorrect. The lower-panel diagnostic bucket contains verifier-passing candidates that the semantic judge did not accept; manually accepted cases in that bucket are not counted as SEMPASS but indicate conservative rejections. Table 8 shows one representative positive audit case.

Audit interpretation. The accepted/rejected manual-validity rates below are pooled within the Reference-Statement and Predicted-Statement audit stages; the undercount and sensitivity columns apply those audited rates to each dataset's evaluation bucket size. The MiniF2F-DSP row reuses the Predicted-Statement accepted-bucket sensitivity because no separate proof-judge-rejected sample was drawn there; its rejected-bucket undercount is therefore zero. The rejected-bucket rates are conditional audit proportions, not dataset-level false-negative rates.

Table 7: Manual audit and post-stratified sensitivity of proof-level semantic evaluation.

Setting	Accepted manual valid	Rejected manual valid	SemPass undercount (pp)	Accepted sensitivity (pp)
LeanWorkbook Reference	94.7% ($n = 150$)	92.5% ($n = 40$)	0.38	2.2–2.9
Herald Reference	94.7% ($n = 150$)	92.5% ($n = 40$)	2.82	2.6–3.4
LeanWorkbook Predicted	98.8% ($n = 80$)	90.0% ($n = 30$)	1.11	0.49
MiniF2F-DSP Predicted	–	–	0.00	0.30

Table 8: Representative semantic audit example from the Reference-Statement positive bucket. The generated Isabelle theory passed Isabelle checking through PISA and was manually accepted under the proof-level semantic audit protocol.

Field	Content
Lean theorem	There exists a function $f : \mathbb{Z} \to \mathbb{Z}$ such that, for all integers x, $f(x) = -x^2$.
Isabelle theorem	There exists a function f :: `int => int` such that $\forall x.\ f\ x = -(x^2)$.
Isabelle proof sketch	The proof instantiates the witness with $\lambda x.\ -x^2$ and closes the goal by simplification.
Audit decision	The Lean and Isabelle statements assert the same integer-valued witness function $x \mapsto -x^2$, and the generated theory is accepted as matching the source theorem under the audit protocol.

C Verifier, Reward, and Decoding Details

All reported proof and end-to-end evaluations use the same Isabelle-through-PISA backend. Table 9 summarizes the train-time and evaluation-time verifier settings. Evaluation uses a shared repair path with the default cheap repair commands—`by auto`, `by simp`, `by blast`, `by fastforce`, `by force`, `by eval`, `by presburger`, `by sos`, `by arith`, `by linarith`, and `by (auto simp: field_simps)`—followed by the evaluation backend's Sledgehammer-enabled repair attempt. The statement- and proof-level semantic-evaluation judges use GPT-5.4 with `reasoning_effort=medium`. During GRPO training, expensive Sledgehammer repair is disabled and the train-time repair set is restricted to cheap methods. Candidate deduplication is enabled throughout.

Table 9: Verifier and decoding settings used for training and evaluation.

Setting	Verifier budget	Decoding	Repair and notes
Train GRPO-MVP	45s completion / 180s task	4 completions; temperature 0.9; top-p 0.9	cheap repair only; Sledgehammer disabled; import-sticky dispatch
Eval Pass@1	shared eval backend	greedy	cheap repair sequence + Sledgehammer-enabled repair; shared by statement, proof, end-to-end, and ablation evaluations
Eval Pass@4	shared eval backend	temperature 0.9; top-p 0.9	same shared evaluation repair path

C.1 Verifier configuration for RL training

The verifier is used in two different roles. During GRPO, it is part of the optimization loop: every batch produces multiple sampled theories, and the verifier must return a stable reward signal quickly enough for policy learning. At evaluation time, the verifier is a measurement backend: it should give every compared method the same opportunity to pass Isabelle checking under the full repair path. We therefore separate the training verifier from the reporting verifier.

The training verifier is intentionally bounded. It uses the same Isabelle-through-PISA execution interface, but applies the completion and task timeouts in Table 9, disables Sledgehammer, and restricts automatic repair to `by auto`, `by blast`, and `by eval`. This setting keeps reward collection tied to proof structure produced by the policy rather than to long and variable external proof search. It also makes failed or malformed completions cheap enough to provide useful negative feedback, which is essential for verifier-guided RL.

Final evaluation uses the shared evaluation backend for all methods. Failed `by ...` steps are repaired with the default cheap method sequence and the evaluation backend's Sledgehammer-enabled repair attempt. Thus the training verifier controls the cost and stability of reward collection, while reported Pass and SemPass numbers are computed with the same full evaluation-time verifier for every baseline, ablation, and final model.

C.2 ADMISSIBILITY, REPAIR, AND MVP TRACE EXTRACTION

During GRPO training, each sampled completion first passes lightweight admissibility gates before it is sent to Isabelle. The normalized completion must contain a complete Isabelle theory block and must not contain uncommented `sorry` or `oops`. Outputs that fail these structural checks receive the configured admissibility penalty and are not treated as verifier successes. In the main GRPO-MVP setting, strict Isabelle success receives reward 1.0, strict verifier failure receives base reward 0.0, invalid structure receives -0.10, and placeholder-containing outputs receive -0.10.

MVP is computed only for failed completions with usable PISA proof-level traces. PISA executes the parsed Isabelle commands and records `proof_level_before` and `proof_level_after` for commands where this information is available. The MVP summarizer excludes the top-level `theory ... begin` command, the final `end`, and empty commands. All remaining parsed Isabelle commands are eligible, including Isar proof commands and method-chain commands when PISA parses them as one command. Positive proof-level deltas contribute to the open magnitude, and negative deltas contribute to the close magnitude. If no raw PISA result exists, no proof-level trace is available, the verifier task is masked by infrastructure failure, or the completion fails before usable proof-level transitions are observed, the MVP term is zero under the main setting. Unfinished proofs are never verifier successes; they receive MVP credit only if PISA executed enough proof-relevant commands to expose open/close transitions before failure.

Table 10 illustrates the extraction on a failed LeanWorkbook-ISA candidate re-run with the same proof-level trace collector used during GRPO. The candidate opens six proof-context levels and closes four before the final `qed` failure, giving $\mathrm{MVP} = 4/6 = 0.667$ and a shaping reward of $0.2 \times 4/6 = 0.133$.

Table 10: MVP trace extraction example. Positive deltas open proof context; negative deltas close proof context. The top-level theory command is excluded from MVP, and the final failing command has no usable after-level.

Step	Isabelle command	Level	Δ	MVP role
1	`theorem lean_workbook_plus_9112: ...`	$0 \to 1$	$+1$	open
2	`proof -`	$1 \to 2$	$+1$	open
3	`have h1: "0 <= (a^2 - 1)^2"`	$2 \to 3$	$+1$	open
4	`by simp`	$3 \to 2$	-1	close
5	`have h2: "0 <= (a^3 - 1)^2"`	$2 \to 3$	$+1$	open
6	`by simp`	$3 \to 2$	-1	close
7	`have h3: "0 <= (a^2 - a)^2"`	$2 \to 3$	$+1$	open
8	`by simp`	$3 \to 2$	-1	close
9	`have h4: "0 <= a^2 * (a - 1)^2"`	$2 \to 3$	$+1$	open
10	`using ha`	$3 \to 3$	0	carry context
11	`by (simp add: ...) nlinarith [...]`	$3 \to 2$	-1	close
12	`qed`	$2 \to \bot$	N/A	failure

D CHECKPOINT AND CURRICULUM DETAILS

Checkpoint selection is performed on the LeanWorkbook-ISA validation split. Table 11 reports the full validation checkpoint table, and Table 12 reports the corresponding test numbers and verifier-call budgets for the equal-budget ablation checkpoint and the final selected checkpoint. The final checkpoint is selected by the fixed validation protocol; it attains the best validation PASS@1 and SEMPASS@1, and ties the best validation SEMPASS@4.

Figure 2 shows how the Difficulty-Ordered curriculum changes the training distribution during the final GRPO-MVP-DifficultyOrdered run. The curriculum panel describes exposure order; it is not an evaluation metric.

D.1 DIFFICULTY-ORDERED CURRICULUM CONTRACT

Table 13 summarizes the frozen signals used to order RL prompts. The curriculum uses coarse bins rather than fine-grained difficulty scores, separating examples by statement-alignment effort,

Table 11: Validation checkpoint sweep for the final GRPO-MVP-DifficultyOrdered run. Metrics are percentages on the LeanWorkbook-ISA validation split with $N = 242$;

Checkpoint	Val PASS@1	Val SEMPASS@1	Val PASS@4	Val SEMPASS@4	Selected
ckpt200	52.1	51.7	58.3	57.4	no
ckpt300	53.3	52.9	59.1	57.9	no
ckpt400	54.1	53.3	59.5	58.7	no
ckpt600	52.9	52.5	60.3	59.1	no
ckpt800	55.0	54.5	59.9	59.9	no
ckpt1000	55.0	54.1	60.7	60.7	no
final	55.8	55.0	61.2	60.7	yes

Table 12: Checkpoint selection summary for the equal-budget ablation checkpoint and the final selected checkpoint. Validation and test metrics are percentages. The notation $\mu \pm \sigma$ indicates the mean and the standard deviation for test SEMPASS@4.

Checkpoint	Val SEMPASS@1 (%)	Val SEMPASS@4 (%)	Test SEMPASS@1 (%)	Test SEMPASS@4 (%)	Verifier calls
GRPO-MVP DifficultyOrdered-300	52.9	57.9	52.3	58.1±1.1	7,428
GRPO-MVP DifficultyOrdered	55.0	60.7	54.3	59.5±0.6	27,097

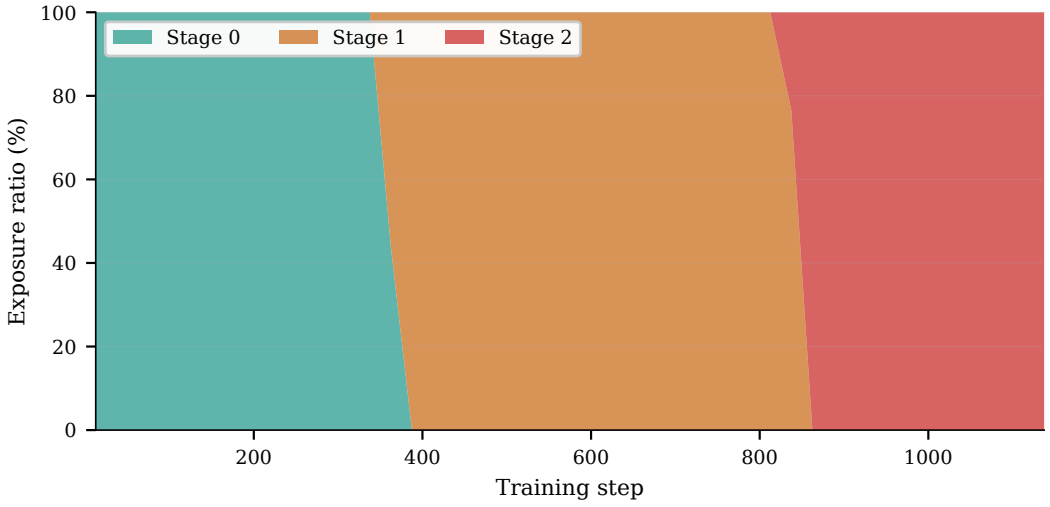


Figure 2: Curriculum exposure for the final GRPO-MVP-DifficultyOrdered run.

source-proof scale, and proof forms such as case analysis, calculation, or induction. These signals are computed before RL and affect only the presentation order of prompts; verifier rewards, MVP shaping, and admissibility penalties are unchanged. The released configuration contains the exact bin boundaries used for the deterministic ordering.

Table 13: Frozen signals used by the Difficulty-Ordered curriculum. The signals define a deterministic presentation order, not reward terms or evaluation metrics.

Signal family	Source	Role in ordering
Statement-alignment effort	first Statement Teacher attempt that both verifies and passes statement-level semantic evaluation	coarse stage assignment
Source proof scale	Lean proof length and theorem length before Isabelle generation	coarse stage assignment
Proof-form tags	presence of structural proof forms such as induction, case analysis, calculational reasoning, and arithmetic/automation-heavy steps	within-stage ordering
Import signature	Isabelle imports required by the accepted target statement	groups nearby prompts to reduce verifier reload churn

E Reproducibility and Compute Disclosure

Table 14 summarizes the main training recipe. The artifact package will include the dataset manifests, training configurations, evaluation scripts, and prompt files needed to reproduce the reported theorem-level translation experiments; the one-time teacher-generation budgets are reported separately below. Statement-SFT and Theory-SFT start from DeepSeek-Prover-V2-7B pretrained weights and are full supervised fine-tunes. GRPO-MVP starts from Theory-SFT and trains a LoRA adapter. The GRPO loss uses TRL's DAPO-style token-level formulation with asymmetric clipping, but online DAPO dynamic group filtering is not used; the reported Difficulty-Ordered condition refers to the input curriculum and import grouping described in Section 3.6.

Table 14: Main model and optimization recipe. We report the settings that define the experimental recipe; lower-level optimizer defaults are left to the released training configuration.

Stage	Training recipe	Reported budget
Statement-SFT	Full fine-tune from DeepSeek-Prover-V2-7B on Herald-ISA statement targets; 1 epoch; LR 2×10^{-5}; bf16	25,581 train examples
Theory-SFT	Full fine-tune from DeepSeek-Prover-V2-7B on Herald-ISA theory targets; 1 epoch; LR 2×10^{-5}; bf16	25,581 train examples
GRPO-MVP	LoRA adaptation from Theory-SFT on LeanWorkbook-ISA; $G = 4$ generations; temperature 0.9, top-p 0.9; LR 5×10^{-6}; $\beta = 0.001$; $\lambda = 0.2$; DAPO-style loss with $\epsilon_{\text{high}} = 0.28$	9,198 train prompts; validation-selected final checkpoint

Table 15 reports training compute and verifier budget for the main student models and RL runs. Verifier calls are unique calls to Isabelle through PISA after candidate deduplication, and all RL rows use $G = 4$ completions per prompt. Student-model training used one RTX PRO 6000 GPU with 96GB memory. The table covers student training and verifier costs; one-time API-assisted teacher and retrieval-index construction budgets are reported with the corresponding construction statistics in Appendix F.

Table 15: Training compute and verifier budget for the main student models and RL runs. N/A indicates stages without RL steps.

Component/run	Train examples	RL steps	Verifier calls	GPU hours	Verifier hours
Statement-SFT	25,581	N/A	0	2.09	0.00
Theory-SFT	25,581	N/A	0	2.60	0.00
GRPO-V DifficultyOrdered-300	9,198	300	7,170	13.67	9.63
GRPO-MVP-Uniform-300	9,198	300	6,948	15.79	7.39
GRPO-MVP DifficultyOrdered-300	9,198	300	7,428	12.83	8.43
GRPO-MVP DifficultyOrdered	9,198	final	27,097	49.29	23.47
GRPO-MVP DifficultyOrdered-λ0.3-300	9,198	300	7,545	12.63	7.98

F Teacher Construction, Retrieval, and SFT Details

Table 16 reports teacher construction statistics. The Statement Teacher constructs LeanWorkbook-ISA target statements, while the Theory Teacher constructs Herald-ISA complete Isabelle theory targets. Retrieval is used only for Theory Teacher construction; the student models do not receive retrieval augmentation at evaluation time.

The retrieval index has a search view and an evidence view. We parse Isabelle/HOL and AFP into structured units; each unit has a natural-language summary u_j for similarity search and formal evidence $e_j = (\mathrm{id}_j, \mathrm{import}_j, \mathrm{code}_j)$. Let $\mathcal{I} = \{(u_j, e_j)\}_{j=1}^{N}$ denote this index. For each `FrenzyMath/Herald_proofs` record, the Theory Teacher queries with the `informal_proof` field, falling back to the natural-language theorem description. Let r be this query and $e(\cdot)$ the embedding model. Retrieval selects

$$\mathcal{R}_K(r) = \mathrm{TopK}_j \ \mathrm{sim}(e(r), e(u_j)) \,. \tag{6}$$

The retrieved identifiers are then mapped back to formal evidence $\mathcal{C}(r) = \{(\mathrm{id}_j, \mathrm{import}_j, \mathrm{code}_j) : j \in \mathcal{R}_K(r)\}$, which is included in the teacher context.

F.1 Supervised initialization losses

Both supervised stages use standard target-token-masked conditional language modeling. For Statement-SFT, the prompt a_i^{st} contains the Lean imports and Lean statement $s_{L,i}$, and the completion is the accepted Isabelle target specification $z_i^\star$. For Theory-SFT, the prompt a_i^{th} contains the Lean imports, Lean proof input x_i, and fixed Isabelle target specification $z_i^\star$, and the completion is the accepted complete Isabelle theory $y_i^\star$. The losses are

$$\mathcal{L}_{\mathrm{stmt}}(\phi) = -\frac{1}{\sum_i |z_i^\star|} \sum_i \sum_{t=1}^{|z_i^\star|} \log q_\phi\big(z_{i,t}^\star \mid a_i^{\mathrm{st}}, z_{i,<t}^\star\big) \,, \tag{7}$$

$$\mathcal{L}_{\mathrm{theory}}(\theta) = -\frac{1}{\sum_i |y_i^\star|} \sum_i \sum_{t=1}^{|y_i^\star|} \log \pi_\theta\big(y_{i,t}^\star \mid a_i^{\mathrm{th}}, y_{i,<t}^\star\big) \,. \tag{8}$$

Only completion tokens are included in the sums; prompt tokens are masked out.

For the Statement Teacher, the *Semantic accept* column is a pre-release candidate count, whereas *Kept* is the example-level release count. The gap therefore combines candidate-to-example canonicalization, source-side duplicate removal, miniF2F-test overlap removal, and final construction filters; Table 6 reports only the explicit evaluation-overlap checks, not an itemized ledger for every pre-release candidate exclusion. Both teacher translation stages use GPT-5.4; the one-time billed-token construction budgets were 10.35M for the Statement Teacher and 567.02M for the Theory Teacher. These API calls are not used by student models at inference or evaluation time.

RAG case study. Table 18 reports aggregate name-use evidence. We additionally show one concrete retrieval case below. The example should be read as evidence that the retrieved Isabelle import and

Table 16: Teacher construction statistics. Raw, Compile success, and Semantic accept count pre-release candidates; Kept counts released examples after canonicalization and final split construction. Avg. cand. is the mean number of generated teacher candidates evaluated per raw source artifact, reported as a construction-effort diagnostic rather than a ratio of the count columns.

Teacher	Raw	Compile success	Semantic accept	Kept	Avg. cand.	Role
Statement Teacher	10,268	10,055	9,789	9,683	1.44	LeanWorkbook statement targets
Theory Teacher	44,553	32,265	26,150	26,104	1.97	Herald-ISA theory targets

Table 17: Retrieval index statistics for Theory Teacher construction. Retrieval is not used by student models at evaluation time.

Metric	Value	Role
Raw Isabelle chunks	319,328	parsed Isabelle/HOL and AFP library units
Chunk-to-NL summaries	319,328	natural-language search view for retrieval
Summary model	`gemini-2.5-flash-lite`	one-time summary-generation model for the search view
Summary billed tokens	1.31B	one-time chunk-to-natural-language summarization budget
Chroma embeddings	310,672	persisted vector records
Embedding model	`text-embedding-v4`	summary-vector embedding model
Embedding dimension	1,024	retrieval embedding size
Vector top-k	20	retrieved items supplied to Theory Teacher construction

Table 18: Examples where retrieved Isabelle material appears in an accepted Theory Teacher output.

Query theme	Retrieved item	Required import	Rank
irreducible divisibility symmetry	`irreducibleD`	`Factorial_Ring`	4
invertible matrix has full rank	`invertible_implies_full_rank`	`Gauss_Jordan.Inverse`	6
prime power divisibility	`aprimedivisor_prime_power`	`Prime_Powers`	1
filter limit under positive multiplication	`filterlim_times_pos`	`HOL.Limits`	5
complex cosine zero criterion	`cos_eq_0`	`Complex_Transcendental`	3
range of a ring homomorphism	`img_is_subfield`	`HOL-Algebra.Subrings`	1

lemma were useful to the Theory Teacher; reported correctness is still determined by Isabelle checking and proof-level semantic evaluation. The query theme is that a unit/invertible square matrix has full rank, and the retrieved item ranked 6th among the supplied Isabelle context items.

Retrieved Isabelle evidence

```
lemma invertible_implies_full_rank:
  fixes A :: "'a::{field}^'n::{mod_type}^'n::{mod_type}"
  assumes inv_A: "invertible A"
  shows "rank A = nrows A"
```

Accepted Theory Teacher output excerpt

```
theory Rank_Of_IsUnit
  imports "Gauss_Jordan.Inverse"
begin

theorem rank_of_isUnit:
  fixes A :: "'a::{field}^'n::{mod_type}^'n::{mod_type}"
  assumes "invertible A"
  shows "rank A = CARD('n)"
  using invertible_implies_full_rank[OF assms]
  unfolding nrows_def by simp

end
```

The accepted output imports `Gauss_Jordan.Inverse` and invokes `invertible_implies_full_rank`, matching the retrieved evidence.

G Proof-Input and Failure Diagnostics

The main text reports the compact proof-input ablation in Table 3. Table 19 gives the original full Pass/SemPass version for both GRPO-MVP-DifficultyOrdered and Theory-SFT, while Table 23 adds the four expanded transforms. Table 20 summarizes coarse failure buckets for the final system.

Table 19: Full Lean input ablation on LeanWorkbook-ISA. Metrics are percentages. The notation $\mu \pm \sigma$ indicates the mean and the standard deviation. The model benefits from Lean-side context and matched proof information.

Input condition	Pass@1↑ (%)	SemPass@1↑ (%)	Pass@4↑ (%)	SemPass@4↑ (%)	N
GRPO-MVP-DifficultyOrdered					
Lean statement + Lean proof + reference Isabelle statement	54.7	54.3	60.1±0.4	59.5±0.6	243
Reference Isabelle statement only (no Lean source)	46.1	45.3	52.2±0.6	51.8±0.7	243
Shuffled Lean proof + reference Isabelle statement	43.6	43.2	50.7±0.7	50.2±0.9	243
Theory-SFT					
Lean statement + Lean proof + reference Isabelle statement	50.6	49.8	56.1±0.9	55.4±0.8	243
Reference Isabelle statement only (no Lean source)	41.6	41.2	49.1±1.1	48.5±0.9	243
Shuffled Lean proof + reference Isabelle statement	39.9	39.5	48.3±1.2	47.9±0.9	243

Table 20 summarizes coarse pass@1 failure taxonomies for the final system. The buckets are constructed from statement semantic-evaluation outcomes, Isabelle checking outcomes, PISA-reported error categories, and proof-level semantic-evaluation outcomes. For Predicted-Statement

LeanWorkbook-ISA, the *verifier pass but not* SEMPASS row reports the full gap between verifier pass and SEMPASS; the proof-failure row is the remaining nonpassing mass after statement semantic-evaluation failures and verifier-pass-but-not-SEMPASS cases are accounted for. The buckets are diagnostic and use the evaluation set as denominator.

Table 20: Compact PASS@1 failure taxonomy for the final system. Rates are percentages and use the evaluation set as denominator; Count is a count.

Setting	Data	Bucket	Count	Rate (%)
Statement	LeanWorkbook-ISA	not accepted by statement semantic evaluation	34	14.0
Statement	MiniF2F-DSP	not accepted by statement semantic evaluation	31	14.3
Reference	LeanWorkbook-ISA	SEMPASS	132	54.3
Reference	LeanWorkbook-ISA	unfinished proof / remaining subgoal	54	22.2
Reference	LeanWorkbook-ISA	missing fact, method, or constant	21	8.6
Reference	LeanWorkbook-ISA	other verifier failure	35	14.4
Reference	LeanWorkbook-ISA	verifier pass but semantic reject	1	0.4
Predicted E2E	LeanWorkbook-ISA	SEMPASS	95	39.1
Predicted E2E	LeanWorkbook-ISA	statement semantic-evaluation failure	34	14.0
Predicted E2E	LeanWorkbook-ISA	remaining proof failure after semantic-evaluation accounting	108	44.4
Predicted E2E	LeanWorkbook-ISA	verifier pass but not SEMPASS	6	2.5
Predicted E2E	MiniF2F-DSP	SEMPASS	52	24.0
Predicted E2E	MiniF2F-DSP	statement semantic-evaluation failure	31	14.3
Predicted E2E	MiniF2F-DSP	proof failure after statement evaluation pass	134	61.8
Predicted E2E	MiniF2F-DSP	verifier pass but semantic reject	0	0.0

Figure 3 gives a complementary first-error profile for verifier-failed outputs in the available construction and evaluation traces. The buckets are heuristic and based only on the first PISA error, so they should be read as diagnostics rather than root-cause annotations. Statement-teacher failures are concentrated in target-specification well-formedness, especially missing type/import context for minimal statement skeletons. In contrast, Theory Teacher construction has a broader mix of type/import, proof-search, missing-fact, and context failures. On LeanWorkbook-ISA Reference-Statement evaluation, both Theory-SFT and GRPO-MVP failures are dominated by unfinished proof search or timeout, which is consistent with proof reconstruction remaining the main bottleneck once a verified target statement is supplied.

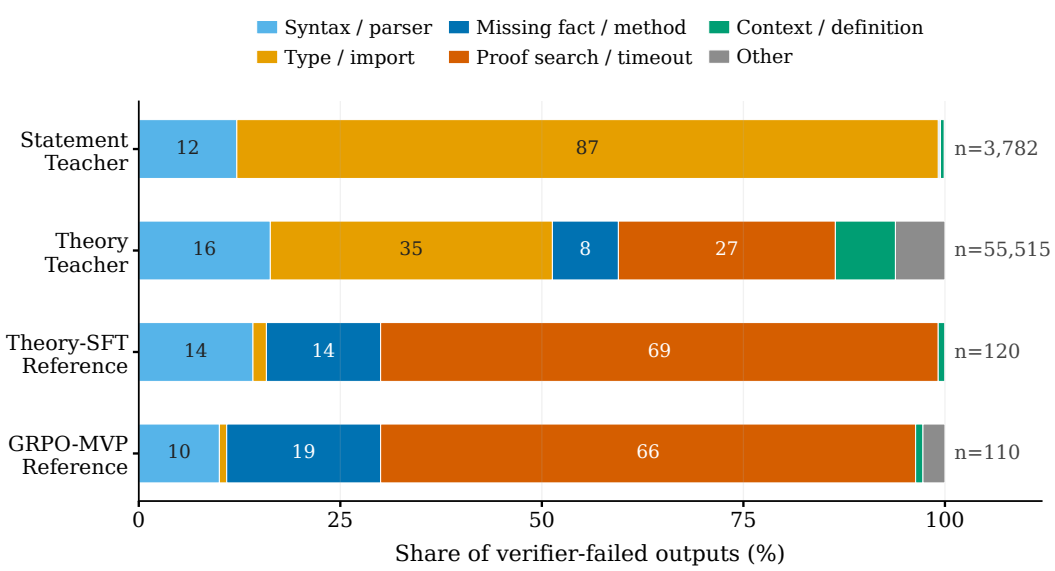


Figure 3: Verifier first-error diagnostics for failed teacher-construction candidate attempts available in the retained construction traces and failed LeanWorkbook-ISA Reference-Statement PASS@1 outputs. Percentages use verifier-failed outputs within each row as denominator; n is the number of failed outputs. Buckets are coarse PISA-message categories and are diagnostic rather than causal.

H STATEMENT-DOMAIN DIAGNOSTICS

The lower Herald-ISA statement-translation score in Table 1 is consistent with the broader and more library-dependent nature of the Herald-derived resource. These diagnostics are descriptive rather than causal: they are computed from the resource files used to construct Herald-ISA and LeanWorkbook-ISA, and they are meant to clarify the domain shift behind the main statement-translation table.

Herald-ISA exposes substantially more source- and target-side context. Among the 26,104 retained Herald-ISA examples, 68.8% contain a usable Lean header; these headers cover 3,777 unique Lean imports, and 87.5% of retained examples have namespace-bearing Lean declaration names. On the Isabelle side, Herald-ISA targets use 829 unique imports, with 34.3% requiring at least one import beyond the base `Main` or `Complex_Main` theories. In 67.4% of examples, the target theory also introduces auxiliary scaffolding before the final theorem statement, such as definitions, locales, records, classes, or helper lemmas. These properties make statement translation depend not only on theorem syntax, but also on cross-library concept mapping and target-context construction.

By contrast, LeanWorkbook-ISA is concentrated around compact workbook-style theorem records. Its extracted Isabelle targets use 38 unique imports, only 4.1% require non-base imports, and 0.5% introduce auxiliary scaffolding; its 90th-percentile target length is also shorter than Herald-ISA (427 versus 781 characters). This concentration helps explain why a statement model trained on broad Herald-ISA supervision transfers well to LeanWorkbook-ISA and MiniF2F-DSP, while Herald-ISA itself remains the harder statement-translation split. MiniF2F-DSP contains compact competition-style Lean proof inputs but no reference Isabelle target statements, so target-side diagnostics are not applicable.

I PROMPT TEMPLATES

This appendix records the main prompt interfaces used in the pipeline. We include the short student prompts in full and provide excerpts for longer teacher, repair, judge, and retrieval-summary prompts. The full prompt files will be released with the artifacts.

I.1 STUDENT MODEL PROMPTS

Statement-SFT prompt

```
Translate the following Lean formal statement into an Isabelle/HOL
    theory file.
Output only Isabelle/HOL code.
The file must start with `theory` and end with `end`.
Use `sorry` as the placeholder proof for every theorem.

### Lean imports
{header}

### Lean formal statement
{formal_theorem}

### Isabelle/HOL theory file
```

Theory-SFT and GRPO proof prompt

```
### Task
Use the Lean proof as guidance to write a complete Isabelle/HOL theory
proving the given Isabelle theorem statement(s).
Output only Isabelle/HOL code.
Do not change the given Isabelle theorem statement(s).
The output must start with `theory`, end with `end`,
and contain no `sorry` or `oops`.

### Lean imports
{header}

### Lean formal proof
{formal_proof}

### Isabelle theorem statement (given)
{formal_statement_norm}

### Isabelle complete theory
```

I.2 TEACHER CONSTRUCTION PROMPTS

For teacher construction, the developer prompt specifies the translation contract and the user prompt supplies the concrete Lean proof or statement. Retrieval context is used only by the Theory Teacher, matching Section 3.3. We include both sides because they define different parts of the interface.

Statement Teacher developer prompt excerpt

```
# Goal
Translate the given Lean statement into an Isabelle/HOL statement and
output a complete Isabelle theory file. This task only requires the
proposition statement; the proof position must use `sorry`.

# Success Criterion
1. The Isabelle/HOL statement is semantically strictly consistent
   with the Lean statement.
2. The output is a complete Isabelle/HOL theory that compiles.

# Authoritative Source Order
1. The Lean formal statement is the only authoritative semantic source.
2. The natural language statement is only for disambiguation and must
```

```
   not override, weaken, or strengthen the Lean formal statement.

# Output Contract
Output only complete Isabelle/HOL theory code. Use only ASCII
    characters.
```

Statement Teacher user prompt

```
### Lean formal statement
{fstmt}

### Natural language statement
{istmt}

## Output Format
Output only complete Isabelle theory code, with no comments,
descriptions, JSON, Markdown, or other wrapping.
```

Theory Teacher developer prompt excerpt

```
# Goal
Translate the given Lean theorem/proof into Isabelle/HOL and generate
Isabelle code that can compile successfully.

# Priority Order
1. Preserve the semantics of the target theorem. Do not change the
   original problem into a definitionally tautological statement in
   order to pass.
2. Maximize the likelihood that Isabelle/HOL code passes compilation.
3. Do not hallucinate theorem names, lemma names, theory names,
   constant names, or import paths.

# Allowed Evidence Policy
Use a non-trivial theorem, lemma, or constant only if it appears in
the retrieved context, is already available in the imports/local
context, or is first locally defined and proved.
```

Theory Teacher user prompt

```
Please migrate the Lean proof into an Isabelle/HOL proof according
to the following input.

# Lean proof
<lean_proof>
{fpf}
</lean_proof>

# Natural language proof
<natural_language_proof>
{ipf}
</natural_language_proof>

# Retrieved Isabelle HOL/AFP library context, which may be used as
    reference
<retrieved_isabelle_context>
{rag_hits}
</retrieved_isabelle_context>
```

I.3 SEMANTIC JUDGE AND REPAIR PROMPTS

EqStmt judge prompt excerpt

```
Given a Lean theorem statement and its Isabelle translation, determine:

same_statement: whether the mathematical semantics of the Isabelle
theorem statement is the same as the Lean theorem statement.

Allow equivalent rewrites, variable renaming, and slight type
specialization, as long as the propositional logical structure is
the same.

Output only JSON:
{
  "same_statement": "yes/no",
  "summary": "Briefly explain the decision."
}
```

Proof semantic judge prompt excerpt

```
Determine whether the given Isabelle theory can be regarded as an
acceptable translation of the Lean theorem.

Judge three dimensions separately:
1. similar_theorem: whether the core proposition proved in Isabelle
   has the same mathematical semantics as the Lean theorem statement.
2. is_trivial: whether the Isabelle proof is structurally trivial.
3. is_acceptable: whether the Isabelle theory is an acceptable
   Lean-to-Isabelle translation.

If similar_theorem = no, then is_acceptable must be no. If the theorem
is preserved and the proof does not pass by tampering with definitions,
weakening the structure, adding unnatural premises, or changing the
problem into a tautology, then is_acceptable is yes.

Output only JSON:
{
  "similar_theorem": "yes/no",
  "is_trivial": "trivial/non-trivial",
  "is_acceptable": "yes/no",
  "summary": "Briefly explain the decision in 1--2 sentences."
}
```

Teacher patch prompt excerpt

```
The full patch prompt has 298 lines. The implementation appends an
error-specific repair section to the original translate prompt.

Error-specific branches include:
Variable not of sort; clash of types; No type arity; Failed to finish
proof; Outer syntax error; Extra variables on rhs; Bad arguments on lhs
    ;
Inner syntax error; Ambiguous input; Illegal reference; Undefined fact;
Undefined locale; Undefined constant; timeout.

### Error Correction - Repair and Compile Isabelle/HOL Theory
Review the compilation error and current theory source. Apply minimal
safe edits that fix all syntax and type errors while preserving working
code. Return the complete corrected Isabelle theory and no extra text.
```

```
### COMPILATION ERROR
{err}

### CURRENT THEORY
{theory_src}
```

RAG chunk-summary prompt excerpt

```
The full RAG chunk-summary prompt has 218 lines. It is used to create
the natural-language search view of Isabelle/HOL and AFP library units.

Translate the given Isabelle material into a textbook-style informal
mathematical write-up. Output only final mathematical prose, with no
Isabelle code, tactic names, or meta-remarks.

If the input contains multiple Isabelle blocks, the last lemma/theorem
is the primary target. Preceding definitions and lemmas are
    dependencies
and should be unfolded only as needed to explain the target.
```

J Qualitative Examples

We include selected qualitative examples to show the granularity of the translation task and the semantic-evaluation checks. These examples are illustrative: they are not additional quantitative evidence and should be read together with the aggregate results in Section 4. Code boxes show selected excerpts, with key lines highlighted in bold.

J.1 Herald-ISA Reference-Statement success

The following Herald-ISA example is evaluated in Reference-Statement mode: the Isabelle statement is supplied, and the proof model generates a complete Isabelle theory. The accepted proof reconstructs divisibility witnesses and explicitly constructs a witness for the target divisibility claim.

Lean proof excerpt: dvd_mul_sub_mul

```
theorem dvd_mul_sub_mul {k a b x y : alpha}
    (hab : k dvd (a - b)) (hxy : k dvd (x - y)) :
    k dvd (a * x - b * y) := by
  convert dvd_add (hxy.mul_left a) (hab.mul_right y) using 1
  rw [mul_sub_left_distrib, mul_sub_right_distrib]
  simp only [sub_eq_add_neg, add_assoc, neg_add_cancel_left]
```

Generated Isabelle theory excerpt

```
theory Dvd_Mul_Sub_Mul
  imports Main
begin

theorem dvd_mul_sub_mul:
  fixes k a b x y :: "'a::comm_ring_1"
  assumes hab: "k dvd (a - b)" and hxy: "k dvd (x - y)"
  shows "k dvd (a * x - b * y)"
proof -
  obtain u where u: "a - b = k * u"
    using hab unfolding dvd_def by blast
  obtain v where v: "x - y = k * v"
```

```
    using hxy unfolding dvd_def by blast
  have "a * x - b * y = k * (u * x + b * v)"
    using u v by (simp add: algebra_simps)
  then show ?thesis
    unfolding dvd_def by blast
qed

end
```

J.2 LeanWorkbook-ISA Predicted-Statement success

The next example is a Predicted-Statement end-to-end success on LeanWorkbook-ISA. The predicted Isabelle statement is accepted by statement semantic evaluation, and the generated proof preserves the interval split in the Lean proof.

Lean proof excerpt: `lean_workbook_plus_29115`

```
theorem lean_workbook_plus_29115 (f g : Real -> Real)
  (h0 : forall x, 0 <= x /\ x <= 1 -> f x = x)
  (h1 : forall x, 1 < x /\ x <= 2 -> f x = x - 2)
  (h2 : forall x, 0 <= x /\ x <= 1 -> g x = -x)
  (h3 : forall x, 1 < x /\ x <= 2 -> g x = -x + 2) :
  forall x, 0 <= x /\ x <= 2 -> f x + g x = 0 := by
  exact fun x hx => by
    rcases hx with <hx1, hx2>
    rcases le_or_lt x 1 with hx3 | hx3
    . simp [h0, h2, hx1, hx3]
    . simp [h1, h3, hx1, hx2, hx3]
```

Generated Isabelle theory excerpt

```
theory Lean_Workbook_Plus_29115
  imports Complex_Main
begin

theorem lean_workbook_plus_29115:
  fixes f g :: "real => real"
  assumes h0: "\<And>x. 0 <= x \<Longrightarrow> x <= 1 \<
    Longrightarrow> f x = x"
    and h1: "\<And>x. 1 < x \<Longrightarrow> x <= 2 \<Longrightarrow>
    f x = x - 2"
    and h2: "\<And>x. 0 <= x \<Longrightarrow> x <= 1 \<Longrightarrow>
     g x = -x"
    and h3: "\<And>x. 1 < x \<Longrightarrow> x <= 2 \<Longrightarrow>
    g x = -x + 2"
  shows "\<forall>x. 0 <= x \<and> x <= 2 \<longrightarrow> f x + g x =
     0"
proof
  fix x
  show "0 <= x \<and> x <= 2 \<longrightarrow> f x + g x = 0"
  proof
    assume hx: "0 <= x \<and> x <= 2"
    then have hx1: "0 <= x" and hx2: "x <= 2" by auto
    show "f x + g x = 0"
    proof (cases "x <= 1")
      case True
      have "f x = x" using h0 hx1 True by blast
      moreover have "g x = -x" using h2 hx1 True by blast
      ultimately show ?thesis by simp
    next
```

```
      case False
      then have hx3: "1 < x" using hx1 by linarith
      have "f x = x - 2" using h1 hx3 hx2 by blast
      moreover have "g x = -x + 2" using h3 hx3 hx2 by blast
      ultimately show ?thesis by simp
    qed
  qed
qed

end
```

J.3 MiniF2F-DSP Predicted-Statement success

This MiniF2F-DSP example is a Predicted-Statement end-to-end success. The proof is long but mechanically readable: it alternates the symmetry assumption with the recurrence assumption to derive a chain of intermediate values.

Lean theorem excerpt: `aime_1988_p8`

```
theorem aime_1988_p8
    (f : Nat -> Nat -> Real)
    (h0 : forall x, 0 < x -> f x x = x)
    (h1 : forall x y, 0 < x /\ 0 < y -> f x y = f y x)
    (h2 : forall x y, 0 < x /\ 0 < y ->
      (x + y) * f x y = y * f x (x + y)) :
    f 14 52 = 364 := by
  -- The Lean proof derives a sequence of intermediate values:
  -- f 2 2, f 2 4, f 4 2, ..., f 14 52.
```

Generated Isabelle theory excerpt

```
theory Aime_1988_P8
  imports Complex_Main
begin

theorem aime_1988_p8:
  fixes f :: "nat => nat => real"
  assumes h0: "\<And>x. 0 < x \<Longrightarrow> f x x = real x"
    and h1: "\<And>x y. 0 < x \<Longrightarrow> 0 < y \<Longrightarrow>
     f x y = f y x"
    and h2: "\<And>x y. 0 < x \<Longrightarrow> 0 < y \<Longrightarrow>
      (real x + real y) * f x y = real y * f x (x + y)"
  shows "f 14 52 = 364"
proof -
  have h3: "f 2 2 = 2" using h0[of 2] by simp
  have h4: "f 2 4 = 4"
  proof -
    have "(real 2 + real 2) * f 2 2 = real 2 * f 2 4"
      using h2[of 2 2] by simp
    then show ?thesis using h3 by simp
  qed
  have h5: "f 4 2 = 4" using h1[of 4 2] h4 by simp
  have h6: "f 4 6 = 12"
  proof -
    have "(real 4 + real 2) * f 4 2 = real 2 * f 4 6"
      using h2[of 4 2] by simp
    then show ?thesis using h5 by simp
  qed
  have h7: "f 4 10 = 20"
  proof -
```

```
    have "(real 4 + real 6) * f 4 6 = real 6 * f 4 10"
      using h2[of 4 6] by simp
    then show ?thesis using h6 by simp
  qed
  have h8: "f 10 4 = 20" using h1[of 10 4] h7 by simp
  have h9: "f 10 14 = 70"
  proof -
    have "(real 10 + real 4) * f 10 4 = real 4 * f 10 14"
      using h2[of 10 4] by simp
    then show ?thesis using h8 by simp
  qed
  have h10: "f 14 10 = 70" using h1[of 14 10] h9 by simp
  have h11: "f 14 24 = 168"
  proof -
    have "(real 14 + real 10) * f 14 10 = real 10 * f 14 24"
      using h2[of 14 10] by simp
    then show ?thesis using h10 by simp
  qed
  have h12: "f 14 38 = 266"
  proof -
    have "(real 14 + real 24) * f 14 24 = real 24 * f 14 38"
      using h2[of 14 24] by simp
    then show ?thesis using h11 by simp
  qed
  have "(real 14 + real 38) * f 14 38 = real 38 * f 14 52"
    using h2[of 14 38] by simp
  then show ?thesis using h12 by simp
qed

end
```

J.4 Statement semantic drift

This example shows a statement semantic-evaluation failure that nevertheless admits a verifier-passing Isabelle proof after the statement has drifted. The Lean theorem distinguishes a real cosine expression from a complex cosine expression; the predicted Isabelle statement collapses both sides to the same real expression. The proof-level semantic judge is skipped in this case because the predicted statement already fails statement semantic evaluation.

Lean source theorem excerpt: `lean_workbook_plus_21632`

```
theorem lean_workbook_plus_21632 (n : Nat) (hn : 2 <= n) :
  (prod k in Icc 1 (n / 2),
     3 + 2 * Real.cos (2 * pi * k / n))
  =
  (prod k in Icc 1 (n / 2),
     3 + 2 * Complex.cos (2 * pi * k / n)) := by
  simp [Finset.prod_congr rfl]
```

Verifier-passing but semantically drifted Isabelle output

```
theory Lean_Workbook_Plus_21632
  imports Complex_Main
begin

theorem lean_workbook_plus_21632:
  fixes n :: nat
  assumes hn: "2 <= n"
  shows "prod (\<lambda>k. 3 + 2 * cos (2 * pi * of_nat k / of_nat n))
           {1..n div 2}
```

```
      =
        prod (\<lambda>k. 3 + 2 * cos (2 * pi * of_nat k / of_nat n))
          {1..n div 2}"
  by simp

end
```

Here Isabelle checking succeeds because the generated statement is reflexive, but the statement is not the same theorem as the Lean source. This example motivates counting predicted-statement failures as end-to-end semantic failures even when a downstream proof can verify the shifted statement.

J.5 Proof-level semantic rejection cases

The proof-level semantic evaluation also rejects verifier-passing Isabelle theories that preserve a superficial target shape while changing the mathematical content. The following three negative cases illustrate representative rejection modes observed in verifier-passing candidates: bypassing the intended proof by introducing ad hoc definitions, replacing the mathematical object with a nearby library object, and weakening the translated setting until the target theorem becomes trivial.

Proof bypass by definition tampering. The Lean theorem below is a library theorem about the free-algebra constructor and lift map.

Lean theorem excerpt: free-algebra lift over injection

```
theorem FreeAlgebra.lift_ι_apply (f : X → A) (x) :
    lift R f (ι R x) = f x := by
  rw [ι_def, lift]
  rfl
```

One verifier-passing Isabelle candidate instead introduces ad hoc definitions inside the generated theory that make the statement true by construction.

Verifier-passing but definition-tampered Isabelle output

```
theory FreeAlgebra_Lift_Iota
  imports Main
begin

type_synonym ('R,'X) free_algebra = 'X

definition iota :: "'R => 'X => ('R,'X) free_algebra" where
  "iota R x = x"

definition lift :: "'R => ('X => 'A) => ('R,'X) free_algebra => 'A"
    where
  "lift R f = f"

lemma FreeAlgebra_lift_iota_apply:
  "lift R f (iota R x) = f x"
  unfolding iota_def lift_def by simp

end
```

This theory checks, but it does not translate the source theorem: it identifies the free algebra with the generator type and defines the injection and lift so that the goal unfolds to a reflexive fact, bypassing the intended source-library semantics. The semantic judge rejects such cases as definition tampering rather than accepted proof translation.

Mathematical-object substitution. The next Lean theorem is about the `Part` monad: mapping a function over a partial value and then binding is equal to binding after applying the function.

Lean theorem excerpt: `Part.bind_map`

```
theorem Part.bind_map {γ} (f : α → β) (x) (g : β → Part γ) :
    (map f x).bind g = x.bind (fun y => g (f y)) := by
  rw [← bind_some_eq_map, bind_assoc]
  simp
```

A verifier-passing Isabelle candidate proves the analogous identity for `Option` instead.

Verifier-passing but object-substituted Isabelle output

```
theory Part_Bind_Map
  imports Main
begin

theorem part_bind_map:
  "Option.bind (map_option f x) g = Option.bind x (%y. g (f y))"
  by (simp add: map_conv_bind_option)

end
```

The Isabelle theorem is valid, but it replaces Lean's `Part` type with Isabelle's `option` datatype without giving a semantics-preserving encoding of partial values. The semantic judge rejects this as a nearby-library substitution rather than a proof translation of the source theorem.

Weakening by extra-assumption trivialization. The next Lean theorem is a category-theoretic statement about cover-dense and locally full functors and Grothendieck topologies. Its proof uses transitivity, pullback stability, functor pushforward image sieves, and downward-closed sieve membership.

Lean theorem excerpt: cover-dense functorPullback/pushforward

```
theorem CategoryTheory.Functor.IsCoverDense.
    functorPullback_pushforward_covering
    [G.IsCoverDense K] [G.IsLocallyFull K] {X : C}
    (T : K (G.obj X)) :
    (T.val.functorPullback G).functorPushforward G ∈ K (G.obj X) := by
  refine K.transitive T.2 _ fun Y iYX hiYX => ?_
  apply K.transitive (G.is_cover_of_isCoverDense _ _) _
  rw [Sieve.pullback_comp]
  apply K.pullback_stable
  apply K.superset_covering ?_
    (G.functorPushforward_imageSieve_mem _ _)
```

A verifier-passing Isabelle candidate collapses the categorical structure to sets and functions, then assumes every sieve collection is universal.

Verifier-passing but trivialized Isabelle output

```
theory Functor_IsCoverDense
  imports Main
begin

locale setting =
  fixes C :: "'c set" and D :: "'d set" and G :: "'c => 'd"
  fixes K :: "'d => 's set"
```

```
  fixes functorPullback :: "('c => 'd) => 's => 's"
  fixes functorPushforward :: "('c => 'd) => 's => 's"
  assumes K_univ: "\<And>y. K y = UNIV"
begin

lemma functorPullback_pushforward_covering:
  fixes X :: "'c"
  fixes T :: "'s"
  assumes "T \<in> K (G X)"
  shows "functorPushforward G (functorPullback G T) \<in> K (G X)"
  using K_univ by simp

end

end
```

The Isabelle lemma is valid under its own assumptions, but it is not a faithful translation: categories become unstructured sets, the functor becomes an ordinary function, the Grothendieck topology is replaced by a set-valued map, and the additional assumption `\<And>y. K y = UNIV` trivializes the goal. This is a verifier pass but a semantic rejection.

K Off-the-Shelf Prover and Automation Diagnostics

Table 21 reports two diagnostic baselines that clarify the Reference-Statement setting. The Isabelle automation baseline is target-statement-only: it receives the supplied Isabelle theorem statement, constructs a statement-only theory, and tries a fixed portfolio of Isabelle methods through the same PISA backend. It does not consume the Lean theorem or Lean proof, and it performs no language-model generation. The result therefore calibrates the fraction of examples already solvable by target-side automation.

We also evaluate DeepSeek-Prover-V2 as an off-the-shelf model prompted to generate Isabelle theories under our statement-conditioned interface, including the LeanWorkbook-ISA Predicted-Statement cascade. This is an interface-transfer diagnostic rather than a native-use benchmark or criticism of DeepSeek-Prover-V2: although our student models initialize from DeepSeek-Prover-V2-7B, the off-the-shelf row omits the Lean-to-Isabelle SFT/RL adaptation. DeepSeek-Prover-V2 is optimized for Lean proof generation, whereas our task requires Isabelle/HOL theory construction, imports, Isar/proof-method idioms, and faithful use of a supplied target statement. Its lower scores should therefore not be read as a weakness of DeepSeek-Prover-V2 in its native Lean setting, but as evidence that Lean-specialized proving ability does not directly transfer to Isabelle/HOL theory construction without interface-specific supervision.

As an additional closed-model calibration, we evaluate GPT-5.4 on LeanWorkbook-ISA with the same SFT/GRPO/evaluation prompts and the same evaluation backend. All GPT-5.4 generation calls in this calibration use `reasoning_effort=medium`; Pass@1 uses temperature 0, while Pass@4 samples four candidates with temperature 0.9 and top-p 0.9. We restrict this calibration to LeanWorkbook-ISA because it is our primary benchmark and is constructed from Lean Workbook traces after miniF2F-test overlap removal. We do not use MiniF2F-DSP for closed-model calibration because public miniF2F formalizations across proof assistants make contamination difficult to rule out for closed APIs. The GPT-5.4 rows should therefore be read as a difficulty calibration for the main interface, not as a controlled multi-benchmark baseline.

Table 21: Diagnostic baselines and closed-model calibration for proof generation. Metrics are percentages. The notation $\mu \pm \sigma$ indicates the mean and the standard deviation. Isabelle automation is target-statement-only and does not use the Lean theorem/proof; DeepSeek-Prover-V2 is evaluated outside its native Lean interface; GPT-5.4 is included only as a LeanWorkbook-ISA closed-model calibration.

Method	Setting	PASS@1↑ (%)	SEMPASS@1↑ (%)	PASS@4↑ (%)	SEMPASS@4↑ (%)	N
LeanWorkbook-ISA						
Isabelle automation	target statement only	37.4	37.4	N/A	N/A	243
DeepSeek-Prover-V2	Reference-Statement	26.7	24.7	32.5±1.1	30.9±1.2	243
DeepSeek-Prover-V2	Predicted-Statement	17.3	12.8	22.1±0.9	18.8±1.1	243
GPT-5.4	Reference-Statement	54.3	52.7	61.9±0.7	61.3±0.6	243
GPT-5.4	Predicted-Statement	37.9	35.0	51.4±1.1	51.0±1.0	243
Herald-ISA						
Isabelle automation	target statement only	50.4	50.4	N/A	N/A	262
DeepSeek-Prover-V2	Reference-Statement	23.7	21.8	31.2±1.3	27.7±1.2	262

L ADDITIONAL DIAGNOSTICS

All tables and plots in this section use the data splits, checkpoints, verifier backend, decoding settings, and evaluation procedures specified in the main paper and the preceding appendices. The training-diagnostic plots are visual summaries of the equal-budget RL conditions and the validation checkpoint sweep defined in Appendix L.3; they do not introduce a separate benchmark or evaluation protocol.

L.1 FROZEN STRUCTURAL BASELINE

The structural baseline reads only `lean.formal_theorem`. A frozen parser and local syntax/symbol replacement rules produce a statement-only Isabelle theory importing `Complex_Main`; for statement-gated outputs we try a fixed 12-method portfolio including `auto`, `simp`, `blast`, `linarith`, and `sledgehammer`. The baseline has no learned parameters, Lean proof input, natural-language input, teacher target, retrieval, or repair loop. Parser, translation, statement, semantic, and automation failures all remain in the full denominator.

Table 22: Full-denominator structural baseline diagnostics.

Dataset	Parser	Translation	Statement Pass	EqStmt	Automation Pass	SemPass
LeanWorkbook-ISA ($N = 243$)	227 (93.4%)	202 (83.1%)	195 (80.2%)	195 (80.2%)	59 (24.3%)	59 (24.3%)
Herald-ISA ($N = 262$)	74 (28.2%)	5 (1.9%)	3 (1.1%)	3 (1.1%)	2 (0.8%)	2 (0.8%)

The most frequent frozen-rule failure codes on Herald-ISA are 62 `unbound_identifier`, 49 `unsupported_character`, 44 multi-name proposition-binder, and 43 typeclass-binder cases. Herald targets use 829 distinct Isabelle imports; 34.3% require imports beyond `Main`/`Complex_Main` and 67.4% introduce definitions, locales, records, classes, or helper lemmas before the final theorem. LeanWorkbook-ISA uses 38 imports, with 4.1% non-base imports and 0.5% scaffolding. These counts diagnose the coverage boundary of this frozen local-rewrite baseline rather than an upper bound on expert symbolic translation.

L.2 Complete source-input ablation

All conditions use the same 243 LeanWorkbook-ISA test IDs, final GRPO-MVP-DifficultyOrdered checkpoint, reference Isabelle statement, greedy decoding, PISA backend, and semantic evaluation. Only the model-visible Lean input changes. "Reference statement only" removes the Lean theorem and proof entirely; "Lean statement-only" retains the Lean declaration and removes only its proof body. Within-proof step shuffle changes 199 examples; 44 have no exchangeable units and remain in the denominator.

Table 23: Full source-input ablation. Metrics are percentages; $\mu \pm \sigma$ denotes the mean and standard deviation over four independent samples.

Input condition	Pass@1	SemPass@1	Pass@4	SemPass@4
Full input	54.7	54.3	60.1±0.4	59.5±0.6
Tactics removed	47.7	46.9	–	–
Within-proof step shuffle	46.9	46.1	–	–
Reference statement only	46.1	45.3	52.2±0.6	51.8±0.7
Proof-skeleton-only	45.3	43.6	–	–
Lean statement-only	44.4	43.2	–	–
Cross-sample shuffled Lean proof	43.6	43.2	50.7±0.7	50.2±0.9

L.3 Equal-budget RL and repair

All equal-budget variants use 300 RL steps and the same LeanWorkbook-ISA test protocol. The curriculum-only comparison is GRPO-MVP-Uniform versus GRPO-MVP-DifficultyOrdered; the latter improves SemPass@1 by 1.7 points. The full selected run uses a validation-selected longer checkpoint.

Table 24: Equal-budget RL ablation on LeanWorkbook-ISA.

Method	Pass@1	SemPass@1	Pass@4	SemPass@4
Theory-SFT	50.6	49.8	56.1±0.9	55.4±0.8
GRPO-V DifficultyOrdered-300	51.4	51.0	57.7±0.8	57.2±0.9
GRPO-MVP-Uniform-300	51.0	50.6	57.9±1.3	57.4±1.3
GRPO-MVP-DifficultyOrdered-300	52.7	52.3	58.9±0.9	58.1±1.1
GRPO-MVP-DifficultyOrdered-λ0.3-300	51.9	51.4	58.1±1.1	57.4±0.8

For inference-only sequential repair, B1 is one generation and B2–B4 receive the preceding theory, PISA's first blocking error, its location, and the current subgoal. Statements are fixed and the semantic judge scores only the final theory. The cumulative trajectory gives Reference SemPass 47.6%, 48.4%, 48.6%, and 48.8% at B1–B4, and Predicted SemPass 35.4%, 36.4%, 36.5%, and 36.6%. Independent Pass@4 is 59.5% (Reference) and 47.5% (Predicted), so the sequential repair gain is concentrated in the first repair round.

L.4 Library-migration interface and component pilot

The theorem-level interface can be extended with a persistent alignment registry for future library-scale migration. The registry schema stores (i) fully qualified Lean declaration IDs, source and target types, namespaces, and class/instance constraints; (ii) source–target abstraction mappings and representation-specific adapters; (iii) Isabelle theory ownership, import DAG, declaration order, and duplicate-link checks; (iv) source-reference edges and target dependency mappings; and (v) verification and semantic-audit status. Resolve reads this state for a new declaration, Translate conditions statement and proof generation on the resolved context, Verify checks the target dependency closure, and Commit adds new entries only after all checks pass. A failed translation leaves the registry unchanged.

As a component-level feasibility check, we assembled a frozen Matrix component containing `Matrix.mul_assoc` and nine declarations whose Lean proofs explicitly invoke it. The joint build verified 10/10 declarations and mapped 9/9 explicit source-reference edges to shared adapters. Packaging changed 10 standalone theories and 11 explicit imports to one shared theory and two imports, and

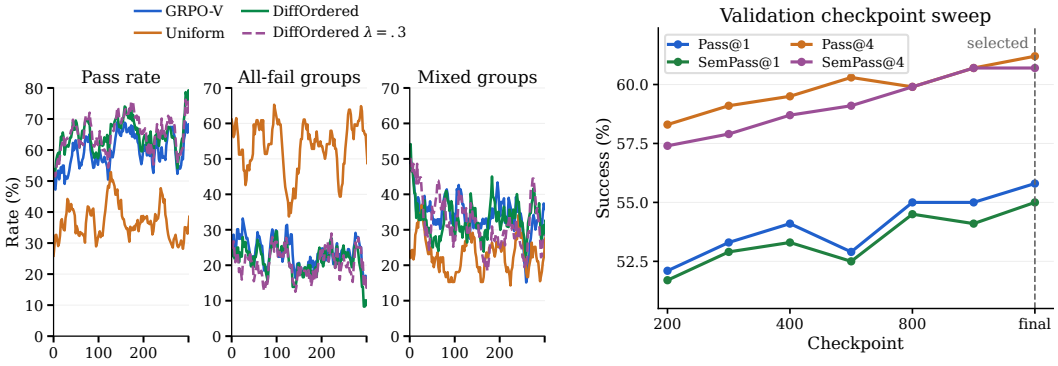


Figure 4: Training diagnostics and validation sweep under the equal-budget RL and checkpoint-selection protocols described above. Left: verifier-pass, all-fail, and mixed-group rates for the equal-budget RL variants. Right: the validation checkpoint sweep for the final GRPO-MVP-DifficultyOrdered run; the selected checkpoint is fixed before test evaluation.

removed two local locale/definition declarations. Two hand-written consumer theories imported the shared component and proved three lemmas without local abstractions. `Matrix.IsHermitian` and `Matrix.conjTranspose` were aligned with native Isabelle concepts `hermitean` and `mat_adj`; two target matrix representations were retained as separate verified adapters. The audit records Lean `NonUnitalCommSemiring` mapping to the stronger Isabelle `comm_semiring_1` as an explicit obligation.

This pilot demonstrates how verified theorem artifacts can expose reusable target interfaces and dependency edges. It does not measure automated registry construction, project-scale graph recovery, or module planning; those are future directions beyond the theorem-level experiments in the main paper.

L.5 Test-time retrieval ablation

The retrieval and control conditions use the same 243 IDs, checkpoint, fp32 weights, greedy decoding, 1,536-token generation budget, PISA backend, and semantic evaluation. Retrieval reuses the Theory Teacher's Isabelle/HOL/AFP index and supplies up to 20 formal evidence items containing an identifier, required import, and code. After inspecting prompt lengths but before viewing test outcomes, we froze a 28,672-token input budget; 222/243 examples retain all 20 items and the remaining 21 retain the ranked prefix.

Table 25: Paired top-20 retrieval ablation on LeanWorkbook-ISA Reference-Statement evaluation.

Condition	Pass@1	SemPass@1
Retrieval-free control	133/243 (54.7%)	132/243 (54.3%)
Top-20 test-time RAG	122/243 (50.2%)	119/243 (49.0%)
RAG − control	−4.5 pp	−5.3 pp

Sample pairing gives 3 fail-to-pass and 14 pass-to-fail transitions for Pass, and 4 fail-to-pass and 17 pass-to-fail transitions for SemPass. The verifier rescues include none of 11 syntax/parser failures, the one type/import failure, or 21 missing-fact/method failures. Among the original 110 verifier failures, first-error inspection separates 53 unresolved-subgoal cases, 9 initial-method failures, 4 unresolved proof states without an explicit error, and 7 explicit timeouts; thus 66/110 (60.0%) are non-timeout proof-construction failures. No generated theory contains an exact retrieved identifier or required-import string, including the rescued cases. This frozen intervention does not rule out jointly trained or interactive retrieval, but it does not support direct top-20 retrieval as the main remedy for the current bottleneck.

L.6 Reference–Predicted pairing and external transfer

The 54.3% Reference-to-39.1% Predicted difference on LeanWorkbook-ISA is a loss of 37 successful IDs. The predicted statement gate fails on 34 IDs (27 Isabelle checking failures and 7 semantic rejections); 22 of these form the net loss. Among the 209 gate-passing IDs, the paired success matrix is:

Table 26: ID-aligned success matrix among the 209 predicted-statement gate-passing examples.

	Predicted success	Predicted failure
Reference success	88	22
Reference failure	7	92

The residual net loss is $22 - 7 = 15$ IDs. Thirty predicted statements equal their references after whitespace normalization and have identical SemPass labels; among the other 179 gate-passing formulations, Reference succeeds on 100/179 and Predicted on 85/179. MiniF2F-DSP contains 217 public Lean-verified proofs, has zero exact normalized formal or natural-language overlap with LeanWorkbook-ISA, and provides no teacher Isabelle target. It reaches 85.7% statement EqStmt@1 and 24.0% end-to-end SemPass@1. This is external-source transfer evidence, not a difficulty-matched estimate of the causal effect of source provenance.

L.7 Source-construction audit

The source audit reads only Lean declaration identities, formal statements, and proofs. Fixed rules assign one primary label for each of 12 domains and nine proof-style categories and never inspect Isabelle targets, teacher outputs, semantic judgments, or final acceptance labels.

Table 27: Mean absolute source-composition shift from raw inputs to retained cohorts.

Dataset	Raw → final	Domain shift (pp)	Proof-style shift (pp)
LeanWorkbook-ISA	10,268 → 9,683	0.12	0.06
Herald-ISA	44,553 → 26,104	1.13	0.90

GPT-5.4 constructs the Isabelle target side; Herald-ISA starts from existing `FrenzyMath/Herald_proofs`, and LeanWorkbook-ISA reconstructs proved Lean-Workbook traces. LeanWorkbook-ISA has no reference Isabelle proof, so scoring does not reward teacher-script reproduction. Every source-present category remains represented after filtering. The audit rules out collapse in these coarse source distributions but cannot exclude within-category selection for concepts, libraries, or target contexts that align better with Isabelle.

L.8 Complete five-axis resource taxonomy

Each retained example receives exactly one primary label on each axis under deterministic fixed-priority rules. The labels are operational proxies, not expert difficulty annotations. MiniF2F-DSP has no reference Isabelle target and is excluded from target dependency and abstraction axes.

The theorem-type axis is reported in Table 32; its classifier removes declaration binders and a top-level premise chain before classifying the main conclusion. The resulting distributions show that LeanWorkbook-ISA is mainly elementary, concrete, and inequality-oriented, whereas Herald-ISA contains broader domains, more equality/characterization theorems, richer proof styles, and substantially more target-side scaffolding.

Table 28: Mathematical-domain distribution.

Domain	LeanWorkbook-ISA	Herald-ISA
Category theory	0 (0.00%)	2,183 (8.36%)
Probability	0 (0.00%)	282 (1.08%)
Number theory	465 (4.80%)	1,402 (5.37%)
Geometry/orientation	0 (0.00%)	1,002 (3.84%)
Analysis/topology/measure	1,349 (13.93%)	3,991 (15.29%)
Linear/tensor algebra	30 (0.31%)	1,795 (6.88%)
Data structures/computation	1 (0.01%)	2,829 (10.84%)
Finite combinatorics	612 (6.32%)	1,780 (6.82%)
Abstract algebra	21 (0.22%)	3,457 (13.24%)
Logic/sets/order	190 (1.96%)	3,375 (12.93%)
Elementary arithmetic/algebra	7,013 (72.43%)	2,869 (10.99%)
Foundations/other	2 (0.02%)	1,139 (4.36%)

Table 29: Primary Lean proof-style distribution.

Proof style	LeanWorkbook-ISA	Herald-ISA
Simplification/rewriting	2,022 (20.88%)	15,378 (58.91%)
Arithmetic/algebraic automation	3,911 (40.39%)	75 (0.29%)
General automation/decision	31 (0.32%)	225 (0.86%)
Direct closure	1,141 (11.78%)	347 (1.33%)
Structured forward/backward reasoning	2,239 (23.12%)	5,197 (19.91%)
Case analysis/induction	205 (2.12%)	2,885 (11.05%)
Calculational/transitivity	123 (1.27%)	76 (0.29%)
Extensionality/congruence	11 (0.12%)	1,879 (7.20%)
Other specialized style	0 (0.00%)	42 (0.16%)

Table 30: Target dependency-depth distribution. Each example receives its highest observable level.

Level	LeanWorkbook-ISA	Herald-ISA
D0: base context only	9,234 (95.36%)	3,804 (14.57%)
D1: non-base direct import	397 (4.10%)	4,706 (18.03%)
D2: scaffolding without parsed multihop chain	28 (0.29%)	13,047 (49.98%)
D3: scaffolding with parsed multihop chain	24 (0.25%)	4,547 (17.42%)

Table 31: Artifact abstraction-level distribution.

Level	LeanWorkbook-ISA	Herald-ISA
A0: concrete fixed type/domain	8,158 (84.25%)	2,791 (10.69%)
A1: parametric/higher-order signature	1,216 (12.56%)	1,732 (6.64%)
A2: polymorphic/typeclass/abstract structure	257 (2.65%)	3,987 (15.27%)
A3: target scaffolding command	52 (0.54%)	17,594 (67.40%)

Table 32: Primary theorem-type distribution.

Theorem type	LeanWorkbook-ISA	Herald-ISA
Existence/construction	727 (7.51%)	498 (1.91%)
Equivalence/characterization	501 (5.17%)	5,128 (19.64%)
Equality/identity	3,228 (33.34%)	14,418 (55.23%)
Negation/disequality	42 (0.43%)	268 (1.03%)
Inequality/order	4,009 (41.40%)	1,816 (6.96%)
Divisibility/congruence	415 (4.29%)	155 (0.59%)
Membership/inclusion	43 (0.44%)	679 (2.60%)
Logical connective/proposition	631 (6.52%)	214 (0.82%)
Arithmetic predicate/property	27 (0.28%)	64 (0.25%)
Function mapping/regularity	35 (0.36%)	830 (3.18%)
Structural/relational property	19 (0.20%)	1,187 (4.55%)
Other predicate/property	6 (0.06%)	847 (3.24%)