Lean2Isabelle: Factorized Cross-Assistant Proof Translation

Guanyu Liu [*1] **Jingkun Ma** [*1] **Derek F. Wong** [1]

Abstract

Formal proofs are spread across proof assistants, but reusing a verified Lean proof in Isabelle/HOL requires more than syntax translation: the target theory must check and preserve the source theorem's meaning under semantic evaluation. We introduce Lean2Isabelle, a verifier-grounded Lean-to-Isabelle proof-translation framework that factorizes the task into target-statement prediction and statement-conditioned Isabelle theory generation. This factorization supports Reference-Statement evaluation, which isolates proof reconstruction with a verified target statement, and Predicted-Statement evaluation, which measures the end-to-end cascade. We construct Herald-ISA and LeanWorkbook-ISA, derive MiniF2F-DSP from public DeepSeek-Prover-V2 miniF2F Lean proofs, and train Statement-SFT, Theory-SFT, and a GRPO proof translator with Isabelle/PISA verifier feedback, Minimal Valid Progress shaping, and a Difficulty-Ordered curriculum. Statement-SFT reaches 86.0% and 85.7% EqStmt@1 on LeanWorkbook-ISA and MiniF2F-DSP; in Reference-Statement proof translation, GRPO-MVP-DifficultyOrdered raises LeanWorkbook-ISA SemPass@1 from 49.8% to 54.3%. In Predicted-Statement translation, the final system reaches 39.1% and 24.0% SemPass@1 on LeanWorkbook-ISA and MiniF2F-DSP, while source ablations lower LeanWorkbook-ISA SemPass@1 to 45.3% without Lean input and 43.2% with a shuffled Lean proof, indicating that matched Lean context carries useful proof signal.

1. Introduction

Formal mathematics is not organized around a single proof assistant. Lean, Isabelle/HOL, Rocq, and other systems use different foundations, libraries, automation, and proof idioms, so a theorem proved in one assistant is not immediately reusable in another. For ML theorem proving, this fragmentation makes each assistant look like a separate learning problem: models cannot directly reuse verified proof artifacts across ecosystems, and evaluations can hide failures that appear only when a proof is moved between foundations and libraries.

Recent neural theorem provers have improved proof search inside individual assistants. Lean systems such as LeanDojo/ReProver, DeepSeek-Prover, Goedel-Prover, and APOLLO combine retrieval, verifier feedback, search, reinforcement learning, synthetic data, tool integration, and repair (Yang et al., 2023; Xin et al., 2025a; Ren et al., 2025; Lin et al., 2025; 2026; Ospanov et al., 2026); Isabelle/HOL systems such as Thor, Draft-Sketch-Prove, Baldur, and Magnushammer show analogous benefits from interaction, proof sketches, repair, and premise selection (Jiang et al., 2022; 2023; First et al., 2023; Mikuła et al., 2024). These systems rely on verifier feedback, but usually assume that the theorem statement and proof environment already live in the same assistant. We instead ask whether an already verified Lean proof can be reused to construct a verifiable Isabelle/HOL proof artifact.

This paper studies verified Lean-to-Isabelle proof translation as the learning problem behind proof-artifact migration. Given a Lean theorem and proof, the system must generate a complete Isabelle/HOL theory accepted by Isabelle checking through PISA and LLM-based semantic evaluation (de Moura & Ullrich, 2021; The mathlib Community, 2020; Nipkow et al., 2002; Jiang et al., 2021). This is not ordinary autoformalization, which starts from informal mathematics, nor same-assistant proving, which assumes a target-side statement and library context. The input is already a verified proof artifact, and accepted success is counted only when the artifact does not shift assumptions, domains, library facts, or theorem strength. We therefore operationalize success through target-system checking together with audited semantic evaluation against the source theorem rather than claiming a machine-checked cross-assistant preservation certificate.

Existing theorem-proving, autoformalization, and verifier-guided learning work supplies important ingredients, but it does not test proof-level migration from a verified source

[*]Equal contribution [1]NLP2CT Lab, University of Macau, Macau, China. Correspondence to: Derek F. Wong <derekfw@um.edu.mo>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

proof to a complete checked target theory. Retrieval, sketch guidance, proof repair, and verifier-based RL help proof search once the target statement and environment are fixed (Yang et al., 2023; Mikuła et al., 2024; Jiang et al., 2023; First et al., 2023; Ospanov et al., 2026; Shao et al., 2024; Guo et al., 2025; Yu et al., 2026; Ren et al., 2025; Ji et al., 2025; Kim & Yun, 2026; Wang et al., 2026b; Hubert et al., 2026). Autoformalization and multi-system benchmarks show that type-checking alone is insufficient for faithful statement translation and motivate semantic consistency, alignment, and dependency-aware evaluation (Wu 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; Zheng et al., 2022; Tsoukalas et al., 2024; Viennot et al., 2025; Stoskopf et al., 2025). The remaining question is where cross-assistant proof translation fails: target-statement alignment, target proof reconstruction, or the cascade between them.

We factor cross-assistant proof reuse into target-statement prediction and statement-conditioned Isabelle theory generation. A statement model maps the Lean theorem statement to an Isabelle/HOL target statement, and a statement-conditioned proof model generates a complete Isabelle theory for that target. Reference-Statement evaluation provides a verified Isabelle statement to isolate proof generation; Predicted-Statement evaluation cascades the statement and proof models to measure end-to-end migration. This separation reports both how well the system reconstructs Isabelle theories when the target statement is fixed and how much success remains when the target statement is predicted.

We build this setup from verified resources and verifier-centered training. Herald-ISA translates complete `FrenzyMath/Herald_proofs` records into 26,104 examples filtered by Isabelle verification and LLM-based semantic evaluation, whose training split supervises Statement-SFT and Theory-SFT (Gao et al., 2025). LeanWorkbook-ISA reconstructs Lean proofs from proved Lean-Workbook tactic traces, removes miniF2F-test overlap, and retains 9,683 verified statement-level examples for RL and main evaluation (Ying et al., 2024; Zheng et al., 2022). MiniF2F-DSP derives proof inputs from public DeepSeek-Prover-V2 miniF2F Lean proofs for end-to-end evaluation (Ren et al., 2025; Zheng et al., 2022). We construct verified statement and complete Isabelle theory teacher targets, use Isabelle/HOL/AFP retrieval only during theory-teacher construction (Nipkow et al., 2002; Archive of Formal Proofs, 2026), train statement and theory models with supervised initialization, and optimize the proof model with Isabelle verifier feedback via PISA, bounded Minimal Valid Progress (MVP) shaping, and a Difficulty-Ordered curriculum.

The experiments separate statement errors from proof-reconstruction errors and test whether Lean proofs add useful information. We ask whether target statements are checkable and semantically faithful, whether the proof model can reconstruct Isabelle theories when the statement is fixed, and how much performance is lost when statement prediction and proof generation are cascaded. We report pass@k, EqStmt@1, and SemPass@k on Herald-ISA, LeanWorkbook-ISA, and MiniF2F-DSP, with equal-budget ablations for verifier-only RL, MVP shaping, and curriculum effects.

Contributions.

- We introduce Lean2Isabelle for Lean-to-Isabelle proof translation; an output is counted successful only after target verification and semantic evaluation against the source theorem.
- We define Reference-Statement and Predicted-Statement evaluation regimes that measure both statement-conditioned proof reconstruction and end-to-end migration.
- We construct derived Lean-to-Isabelle resources, including Herald-ISA for supervised initialization and LeanWorkbook-ISA for RL and main evaluation, and derive MiniF2F-DSP for end-to-end evaluation on miniF2F-style proof inputs; we provide a compact supplementary artifact with sanitized samples and reproducibility metadata, and plan to make Herald-ISA and LeanWorkbook-ISA publicly available once the remaining publication and release constraints are resolved.
- We provide a verifier-grounded empirical characterization of early Lean-to-Isabelle migration: semantic statement checking exposes drift beyond compilation, verifier-guided RL yields consistent numerical gains in statement-conditioned theory reconstruction, and source ablations show that Lean source context and matched Lean proofs provide signal beyond target statements.

2. Related Work

LLMs for formal theorem proving. Learning-assisted theorem proving is largely organized around a single assistant. Lean work scales proof generation through retrieval, verifier feedback, search, synthetic data, RL, tool use, and repair (The mathlib Community, 2020; Yang et al., 2023; Xin et al., 2025a; Ren et al., 2025; Wang et al., 2025; Lin et al., 2025; 2026; Xin et al., 2025b; Ji et al., 2025; Wang et al., 2026b; Ospanov et al., 2026). Isabelle/HOL work builds on Isabelle/HOL, Isar, locales, Sledgehammer, AFP, LISA/PISA, Thor, Draft-Sketch-Prove, Baldur, Magnushammer, and Isabellm for programmatic interaction, ATP integration, sketch guidance, repair, premise selection, and AFP-

backed proof search (Nipkow et al., 2002; Wenzel, 1999; Ballarin, 2014; Böhme & Nipkow, 2010; Archive of Formal Proofs, 2026; Jiang et al., 2021; 2022; 2023; First et al., 2023; Mikuła et al., 2024; Hou, 2026). Our setting shares the verifier-centered techniques but not the same-assistant assumption: it starts from a verified Lean proof and reconstructs both an Isabelle target specification and a complete Isabelle theory.

Autoformalization and semantic faithfulness. Autoformalization converts informal mathematics into formal statements or proofs, and recent work increasingly evaluates faithfulness rather than mere type-checking. miniF2F, ProofNet, PutnamBench, Lean Workbook, and Herald provide formal-mathematics resources (Zheng et al., 2022; Azerbayev et al., 2023; Tsoukalas et al., 2024; Ying et al., 2024; Gao et al., 2025); FormalAlign, symbolic-equivalence and semantic-consistency checks, BEq, ASSESS-style similarity, dependency graphs, retrieval, abstraction learning, and semantic/legal filters highlight that a valid formal statement can still shift the intended mathematics (Wu et al., 2022; 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; Wang et al., 2026b). We inherit this semantic-faithfulness requirement, but evaluate proof-level cross-assistant translation after the target Isabelle statement is fixed or predicted. Related AI-for-math work studies complementary inputs to formal or mathematical reasoning, including formalization-oriented informal proof optimization and visual-aided mathematical reasoning benchmarks (Ma et al., 2026a;b); our focus is proof-artifact migration after the source proof has already been verified.

Verifier-guided learning. Verifier-guided learning and RLVR are common in mathematical reasoning, code, and formal proof generation. DeepSeekMath, DeepSeek-R1, DAPO, PRM800K, Rewarding Progress, and PRIME study GRPO-style or process-oriented learning from verifiable outcomes (Shao et al., 2024; Guo et al., 2025; Yu et al., 2026; Lightman et al., 2024; Setlur et al., 2025; Cui et al., 2025). Formal-proving systems such as DeepSeek-Prover-V1.5/V2, Leanabell-Prover-V2, AlphaProof, GAR, Goedel-Prover-V2, LongCat-Flash-Prover, and Process-Verified RL use proof-assistant feedback for search, RL, multi-turn correction, subgoal decomposition, or tactic-level process supervision (Xin et al., 2025a; Ren et al., 2025; Ji et al., 2025; Hubert et al., 2026; Wang et al., 2026c; Lin et al., 2026; Wang et al., 2026b; Kim & Yun, 2026). GAR further introduces an implicit curriculum that is difficulty-aligned with the prover's evolving capability (Wang et al., 2026c). Our use of Isabelle feedback through PISA follows this paradigm within a cross-assistant interface whose correctness also depends on statement preservation.

Cross-system formalization and proof translation. Cross-system interoperability includes multi-system benchmarks such as miniF2F, PutnamBench, NTP4VC, and ProofGym (Zheng et al., 2022; Tsoukalas et al., 2024; Xu et al., 2026a; Li et al., 2025), as well as interchange, export, and concept-alignment efforts such as Dedukti, Logipedia, proof-assistant library export, and library alignment (Assaf et al., 2023; Dowek & Thiré, 2023; Kohlhase & Rabe, 2021; Gauthier & Kaliszyk, 2019). Recent translation work covers adjacent artifact boundaries: theorem-statement translation in MiniF2F-Rocq, miniF2F translation to Dafny, Lean–Rocq proof-term translation, and simplified Isabelle target languages for neural proving (Viennot et al., 2025; Baksys et al., 2025; Stoskopf et al., 2025; Xu et al., 2026b). Our boundary is different: the input is a verified Lean theorem/proof artifact, the output is a complete Isabelle/HOL theory file, and evaluation separates fixed-target proof reconstruction from the predicted-statement cascade.

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

Figure 1 summarizes the pipeline. Verified teacher targets support retrieval-free Statement-SFT and Theory-SFT models; the proof model is then optimized 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 semantically faithful. We retain statement targets only when they check inside a minimal Isabelle theory and pass statement-level LLM-based semantic

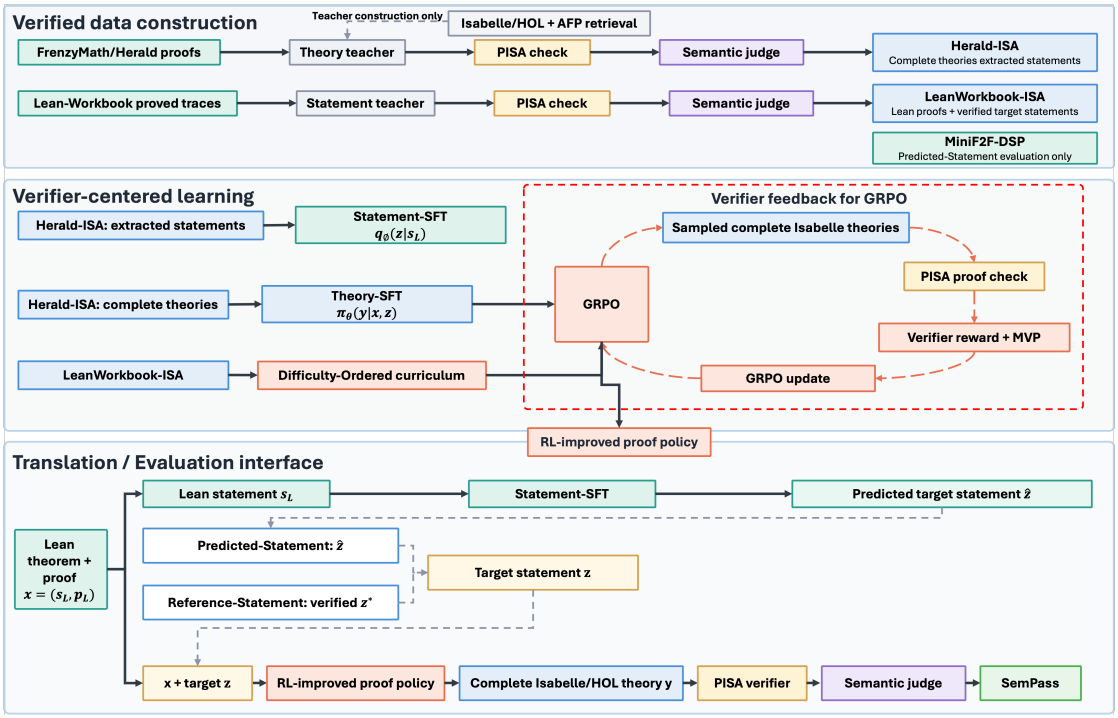

Figure 1. Overview of the verified Lean-to-Isabelle framework: verified data construction, statement/proof models, Isabelle/PISA verifier feedback, and Reference-Statement versus Predicted-Statement evaluation.

evaluation; we retain complete theories only when the full Isabelle theory verifies and passes proof-level LLM-based semantic evaluation. The semantic judges reject domain shifts, missing or additional assumptions, changed goals, unnatural premises, or proofs of a different result. Because these paired resources are constructed through teacher translation, verifier checking, and semantic filtering rather than sampled from naturally occurring Lean–Isabelle parallel corpora, we use them to diagnose where translation fails: target statement, proof reconstruction, or their composition. We use *semantic faithfulness* for the intended mathematical preservation property and *semantic evaluation* for this operational filter; Appendix A gives the formal filters, and Section 4.1 audits the proof-level judge.

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 for Theory Teacher construction, where complete Isabelle theories must rely on target-library definitions, lemmas, theorem statements, and imports (Nipkow et al., 2002; Archive of Formal Proofs, 2026). It supplies target-library evidence to the teacher for avoiding nonexistent lemma names or missing imports; Statement Teacher and all student models remain retrieval-free at evaluation time.

Table 1. Main data domains and their roles. Counts are examples after the construction pipeline in Section 3.2; filtering details and overlap checks are expanded in Appendix A.

Domain	Source	Kept	Train	Val	Test	Main role
Herald-ISA	`FrenzyMath/Herald_proofs`	26,104	25,581	261	262	SFT supervision and Reference-Statement evaluation
LeanWorkbook-ISA	proved Lean-Workbook tactic traces	9,683	9,198	242	243	RL training, main Reference-Statement evaluation, and Predicted-Statement evaluation
MiniF2F-DSP	DeepSeek-Prover-V2 miniF2F Lean proofs	217	N/A	N/A	217	miniF2F-derived Predicted-Statement transfer evaluation

The retrieval index has a search view and an evidence view. We parse Isabelle/HOL and AFP units into natural-language summaries for similarity search and linked formal evidence containing theorem identifiers, required imports, and code snippets. For each `FrenzyMath/Herald_proofs` record, the Theory Teacher queries with the `informal_proof` field, falling back to the natural-language theorem description, and receives the top-k formal evidence. Final correctness is still determined by Isabelle/PISA checking and proof-level semantic evaluation; Appendix F gives the formal retrieval definition and diagnostics.

3.4. Supervised initialization

Supervised initialization teaches the student models to produce well-formed Isabelle/HOL outputs 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 and no admissibility penalty; 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

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

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:

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

where $R_{\text{adm}} \leq 0$ denotes admissibility penalties that are zero for admissible verifier-passing outputs, $\text{adm}(c_i)$ means structurally admissible, and $\lambda < 1$ bounds failed-completion shaping below strict success. Since $0 \leq \mathrm{MVP}(c) \leq 1$, an

admissible 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:

$$M = \sum_{i,t} m_{i,t}, \qquad s_{i,t} = \min(\rho_{i,t}A_i,\ \bar{\rho}_{i,t}A_i)\,,$$
$$\mathcal{L}(\theta) = -\frac{1}{M}\sum_{i,t} m_{i,t}s_{i,t} + \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \tag{3}$$
$$\bar{\rho}_{i,t} = \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\,.$$

The objective combines GRPO-style group-relative advantages with DAPO-inspired token aggregation, generated-token normalization, and asymmetric clipping (Yu et al., 2026); the KL term is estimated over generated tokens under the same mask. 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

The experiments ask three questions about Lean-to-Isabelle proof translation: whether statement drift is measurable, whether proof reconstruction improves with verifier-guided RL, and whether Lean source inputs provide signal beyond the target Isabelle statement. 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 report pass@k, EqStmt@1, and SemPass@k. EqStmt@1 adds statement-level semantic evaluation. SemPass@k combines Isabelle verification with LLM-based proof-level semantic evaluation over verifier-passing theories, following recent formalization pipelines that pair formal checks with LLM semantic evaluation (Lin et al., 2026). The proof-level judge first checks theorem similarity between the Lean theorem and the Isabelle theory before deciding acceptability; in Predicted-Statement evaluation, the predicted statement must also pass the statement evaluation before the downstream proof candidate can count as SemPass. We interpret SemPass as an audited semantic-evaluation metric rather than a machine-checked semantic-preservation certificate. Table 2 gives a compact audit of judge-accepted verifier-passing candidates; Table 10 in Appendix B reports the full audit, including conservative rejections. Verifier settings and judge prompts are deferred to Appendices C and I.

Table 2. Compact audit of judge-accepted verifier-passing candidates. Acceptance and FPR upper bound are percentages; full audit details are in Appendix B.

Setting	n	Manual accepted (%)	FPR upper bound (%)↓
Reference-Statement	150	94.7	5.3
Predicted-Statement	80	98.8	1.3

4.2. Statement translation

Statement translation should expose semantic drift rather than just compilation failures. 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.

Table 3. 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

4.3. Reference-Statement proof translation

Reference-Statement evaluation asks whether verifier-guided learning improves proof reconstruction once statement translation is removed from the loop. 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 feedback yields further numerical gains.

The Reference-Statement block of Table 4 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 numerical gains over Theory-SFT on both LeanWorkbook-ISA and Herald-ISA. This supports the more limited claim that verifier-guided learning is useful for proof reconstruction after statement translation is removed from the loop.

4.4. Equal-budget RL ablation

The RL ablation tests whether the observed gain is associated with verifier-structured learning rather than extra generation budget. Under the same 300-step LeanWorkbook-ISA budget, verifier-only Difficulty-Ordered training raises SemPass@1 from 49.8% to 51.0%, MVP with uniform input ordering reaches 50.6%, and MVP with Difficulty-Ordered input ordering reaches 52.3%; increasing the MVP weight to $\lambda = 0.3$ lowers the 300-step score to 51.4%. The final GRPO-MVP-DifficultyOrdered model in Table 4 uses the same recipe with a longer validation-selected checkpoint. Figure 2 in the appendix visualizes the corresponding verifier-feedback rates and validation sweep.

Table 5. Equal-budget RL ablation on LeanWorkbook-ISA. Metrics are percentages; all RL variants use 300 steps. Appendix D gives the complete pass@k table and validation sweep.

Method	Calls	SemPass@1↑ (%)	SemPass@4↑ (%)
Theory-SFT	0	49.8	55.4±0.8
GRPO-V-DO-300	7,170	51.0	57.2±0.9
GRPO-MVP-Uniform-300	6,948	50.6	57.4±1.3
GRPO-MVP-DO-300	7,428	52.3	58.1±1.1
GRPO-MVP-DO-λ0.3-300	7,545	51.4	57.4±0.8

4.5. Predicted-Statement end-to-end translation

Predicted-Statement evaluation measures the full translation cascade with predicted target Isabelle statements. The lower block of Table 4 separates two effects: GRPO still yields numerical gains after statement prediction, but the cascade loses success when the statement stage drifts. On LeanWorkbook-ISA, GRPO-MVP-DifficultyOrdered reaches 39.1% SemPass@1 and 47.5% SemPass@4, raising scores 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, raising scores over Theory-SFT by +3.3 and +3.8 points. These are end-to-end migration scores rather than proof-model capacity alone; MiniF2F-DSP is a transfer diagnostic on miniF2F-derived Lean proof inputs, not a same-assistant Lean proving comparison.

4.6. Does the model use the Lean proof?

The proof-input ablation tests whether 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 6).

Table 6. Lean proof input ablation on LeanWorkbook-ISA ($N = 243$). Full input denotes the Lean statement, matched Lean proof, and reference Isabelle statement.

Input condition	GRPO SemPass@1	GRPO SemPass@4	Theory-SFT SemPass@1	Theory-SFT SemPass@4
Full input	54.3	59.5±0.6	49.8	55.4±0.8
Reference statement only	45.3	51.8±0.7	41.2	48.5±0.9
Shuffled Lean proof	43.2	50.2±0.9	39.5	47.9±0.9

These ablations separate target-statement solvability from Lean-side signal. Reference-only remains nontrivial because the Isabelle statement exposes the target goal and types, but the full-input gains show that Lean context helps. Shuffling lowers performance further, suggesting that the matched proof contributes theorem-specific signal.

The same comparison also helps interpret the remaining end-to-end gap. Some examples can be solved from the target statement and the learned Isabelle prior alone, but the matched Lean proof supplies useful intermediate structure when the theorem crosses libraries and proof idioms. Conversely, an unrelated proof is worse than removing the Lean side entirely, suggesting that extra source-side tokens are not automatically beneficial. We therefore use this ablation as evidence of conditional source-proof use, while keeping the success criterion fixed: generated theories must still pass Isabelle checking and proof-level semantic evaluation.

Table 4. 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

5. Limitations

Three limitations matter for interpreting these results. First, the paired resources are constructed through teacher translation, verifier checking, and semantic filtering, so they are controlled diagnostic benchmarks rather than natural parallel corpora. Second, SemPass is stricter than verifier pass rate but is not a machine-checked source-preservation certificate: LLM semantic evaluation can reject valid translations or miss subtle premise, domain, or library-definition shifts, so we also report verifier pass rates. Third, proof reconstruction remains the bottleneck; retrieval-free models may struggle with unfamiliar Isabelle lemmas, imports, or proof idioms, and MVP is only a bounded proof-context proxy. The interface is theorem-level; project-scale migration also requires dependency selection, namespace reconciliation, and target-library design across files.

6. Conclusion

We framed Lean-to-Isabelle proof translation as a verifier-grounded cross-assistant task with separate statement and proof evaluation. Semantic statement checking detects drift beyond compilation, verifier-guided learning yields consistent gains in theory reconstruction, and Lean-side inputs provide signal beyond the target Isabelle statement. The resulting resources, evaluation regimes, and accepted translations provide a checked setup for studying proof-artifact migration.

Impact Statement

This work aims to improve the verification and reuse of formal mathematical artifacts across proof assistants. The main expected benefit is more reliable evaluation and migration of machine-generated formal proofs. Possible risks include over-trusting automatically translated proofs or semantic-evaluation judgments; we mitigate this by requiring target-assistant checking, reporting semantic-evaluation audits, and explicitly distinguishing SemPass from a machine-checked cross-assistant preservation certificate.

Acknowledgements

This work was supported in part by the Science and Technology Development Fund of Macau SAR (Grant Nos. FDCT/0007/2024/AKP, EF2024-00185-FST), the UM and UMDF (Grant Nos. MYRG-GRG2024-00165-FST-UMDF, MYRG-GRG2025-00236-FST), the Tencent AI Lab Rhino-Bird Research Program (Grant No. EF2023-00151-FST), the Dr. Stanley Ho Medical Development Foundation (Grant No. SHMDF-AI/2026/001), and the National Natural Science Foundation of China (Grant No. 62266013).

References

Archive of Formal Proofs. Archive of Formal Proofs, 2026. URL https://www.isa-afp.org/. Accessed: 2026-05-02.

Assaf, A., Burel, G., Cauderlier, R., Delahaye, D., Dowek, G., Dubois, C., Gilbert, F., Halmagrand, P., Hermant, O., and Saillard, R. Dedukti: a logical framework based on the $\lambda\Pi$-calculus modulo theory, 2023. URL https://arxiv.org/abs/2311.07185.

Azerbayev, Z., Piotrowski, B., Schoelkopf, H., Ayers, E. W., Radev, D., and Avigad, J. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics, 2023. URL https://arxiv.org/abs/2302.12433.

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

Ballarin, C. Locales: A module system for mathematical theories. *Journal of Automated Reasoning*, 52(2):123–153, 2014. doi: 10.1007/s10817-013-9284-7.

Böhme, S. and Nipkow, T. Sledgehammer: Judgement day. In *Automated Reasoning*, volume 6173 of *Lecture Notes in Computer Science*, pp. 107–121. Springer, 2010. doi: 10.1007/978-3-642-14203-1_9.

Cabral, R. M., Do, T. M., Yu, X., Tai, W. M., Feng, Z., and Shen, X. 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.

Cui, G., Yuan, L., Wang, Z., Wang, H., Zhang, Y., Chen, J., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., Yuan, J., Chen, H., Zhang, K., Lv, X., Wang, S., Yao, Y., Han, X., Peng, H., Cheng, Y., Liu, Z., Sun, M., Zhou, B., and Ding, N. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

de Moura, L. and Ullrich, S. 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.

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

First, E., Rabe, M. N., Ringer, T., and Brun, Y. 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.

Gao, G., Wang, Y., Jiang, J., Gao, Q., Qin, Z., Xu, T., and Dong, B. Herald: A natural language annotated Lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Se6MgCtRhz.

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

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Hou, Z. Vibe coding an LLM-powered theorem prover, 2026. URL https://arxiv.org/abs/2601.04653.

Hubert, T., Mehta, R., Sartran, L., Horváth, M. Z., Žužić, G., Wieser, E., Huang, A., Schrittwieser, J., Schroecker, Y., Masoom, H., et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. doi: 10.1038/s41586-025-09833-y.

Ji, X., Liu, Y., Wang, Q., Zhang, J., Yue, Y., Shi, R., Sun, C., Zhang, F., Zhou, G., and Gai, K. Leanabell-Prover-V2: Verifier-integrated reasoning for formal theorem proving via reinforcement learning, 2025. URL https://arxiv.org/abs/2507.08649.

Jiang, A. Q., Li, W., Han, J. M., and Wu, Y. 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.

Jiang, A. Q., Li, W., Tworkowski, S., Czechowski, K., Odrzygóźdź, T., Miłoś, P., Wu, Y., and Jamnik, M. Thor: Wielding hammers to integrate language models and automated theorem provers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 8360–8373, 2022.

Jiang, A. Q., Welleck, S., Zhou, J. P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., and Wu, Y. 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.

Kim, M. and Yun, S.-Y. 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.

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

Li, X., Ma, W., Bi, H., Li, Z., Si, X., and Yang, K. 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.

Li, Z., Wu, Y., Li, Z., Wei, X., Yang, F., Zhang, X., and Ma, X. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *Advances in Neural Information Processing Systems*, volume 37, pp. 53598–53625, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Lin, Y., Tang, S., Lyu, B., Wu, J., Lin, H., Yang, K., Li, J., Xia, M., Chen, D., Arora, S., and Jin, C. Goedel-Prover: A frontier model for open-source automated theorem proving. In *Proceedings of the 2nd Conference on Language Modeling (COLM 2025)*, 2025. URL https://openreview.net/forum?id=x2y9i2HDjD.

Lin, Y., Tang, S., Lyu, B., Yang, Z., Chung, J.-H., Zhao, H., Jiang, L., Geng, Y., Ge, J., Sun, J., Wu, J., Gesi, J., Lu, X., Acuna, D., Yang, K., Lin, H., Choi, Y., Chen, D., Arora, S., and Jin, C. 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.

Liu, Q., Zheng, X., Lu, X., Cao, Q., and Yan, J. 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.

Liu, X., Zhu, T., Dong, Z., Liu, Y., Guo, Q., Liu, Z., Chen, Y., and Luo, T. 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.

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

Ma, J., Wang, Y., Huo, Y., and Wong, D. F. FIPO-Prover: Formalization-oriented informal proof optimization for efficient formal theorem proving. In *3rd AI for Math Workshop: Toward Self-Evolving Scientific Agents*, 2026a. URL https://openreview.net/forum?id=3qrIRa4OAw.

Ma, J., Zhan, R., Li, Y., Sun, D., Chan, H. P., Chao, L. S., and Wong, D. F. VisAidMath: Benchmarking visual-aided mathematical reasoning. In *The 64th Annual Meeting of the Association for Computational Linguistics*, 2026b. URL https://openreview.net/forum?id=QY3djsyQOw.

Mikuła, M., Tworkowski, S., Antoniak, S., Piotrowski, B., Jiang, A. Q., Zhou, J. P., Szegedy, C., Kuciński, Ł., Miłoś, P., and Wu, Y. Magnushammer: A transformer-based approach to premise selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oYjPk8mqAV.

Min, M. J., Gao, Y., Sy, W., Li, Z., Si, X., and Bastani, O. 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.

Nipkow, T., Wenzel, M., and Paulson, L. C. (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.

Ospanov, A., Farnia, F., and Yousefzadeh, R. APOLLO: Automated LLM and Lean collaboration for advanced formal reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=fxDCgOruk0.

Ren, Z. Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., Wu, Z. F., Gou, Z., Ma, S., Tang, H., Liu, Y., Gao, W., Guo, D., and Ruan, C. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition, 2025. URL https://arxiv.org/abs/2504.21801.

Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. 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.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Stoskopf, T., Cohen, C., and Tabareau, N. 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.

Tsoukalas, G., Lee, J., Jennings, J., Xin, J., Ding, M., Jennings, M., Thakur, A., and Chaudhuri, S. PutnamBench: Evaluating neural theorem-provers on the putnam mathematical competition. In *Advances in Neural Information Processing Systems*, volume 37, pp. 11545–11569, 2024.

Viennot, J., Baudart, G., Gallego Arias, E. J., and Lelarge, M. 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.

Wang, H., Unsal, M., Lin, X., Baksys, M., Liu, J., Dos Santos, M., Sung, F., Vinyes, M., Ying, Z., Zhu, Z., Lu, J., de Saxcé, H., Bailey, B., Song, C., Xiao, C., Zhang, D., Zhang, E., Pu, F., Zhu, H., Liu, J., Bayer, J., Michel, J., Yu, L., Dreyfus-Schmidt, L., Tunstall, L., Pagani, L., Machado, M., Bourigault, P., Wang, R., Polu, S., Barroyer, T., Li, W.-D., Niu, Y., Fleureau, Y., Hu, Y., Yu, Z., Wang, Z., Yang, Z., Liu, Z., and Li, J. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning, 2025. URL https://arxiv.org/abs/2504.11354.

Wang, H., Xie, R., Wang, Y., Gao, G., XintaoYu, and Dong, B. Aria: an agent for retrieval and iterative autoformalization via dependency graph. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=CPxZClPMiy.

Wang, J., Zhang, J., Guo, Q., Guo, L., Li, R., Zhang, C., Peng, C., Wang, C., Zhao, D., Shi, J., Wang, J., Feng, L., Shen, M., Li, Q., An, S., Wang, S., Shi, W., Xi, X., Li, X., Cao, X., Lu, Y., Zhao, Y., Chen, Z., Lin, Z., Wang, W., Pei, P., and Cai, X. LongCat-Flash-Prover: Advancing native formal reasoning via agentic tool-integrated reinforcement learning, 2026b. URL https://arxiv.org/abs/2603.21065.

Wang, R., Yao, J., Pan, R., Diao, S., and Zhang, T. 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.

Wenzel, M. Isar – a generic interpretative approach to readable formal proof documents. In *Theorem Proving in Higher Order Logics*, volume 1690 of *Lecture Notes in Computer Science*, pp. 167–183. Springer, 1999. doi: 10.1007/3-540-48256-3_12.

Wu, Y., Jiang, A. Q., Li, W., Rabe, M. N., Staats, C., Jamnik, M., and Szegedy, C. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32353–32368, 2022.

Xin, H., Ren, Z. Z., Song, J., Shao, Z., Zhao, W., Wang, H., Liu, B., Zhang, L., Lu, X., Du, Q., Gao, W., Zhang, H., Zhu, Q., Yang, D., Gou, Z., Wu, Z. F., Luo, F., and Ruan, C. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=I4YAIwrsXa.

Xin, R., Xi, C., Yang, J., Chen, F., Wu, H., Xiao, X., Sun, Y., Zheng, S., and Ding, M. BFS-Prover: Scalable best-first tree search for LLM-based automatic theorem proving. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32588–32599, Vienna, Austria, 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1565.

Xu, Q., Luan, X., Wang, R., Leang, J. O. J., Wang, P., Li, H., Li, W., and Watt, C. 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.

Xu, Q., Wang, R., Wang, P., Li, H., and Watt, C. 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.

Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R. J., and Anandkumar, A. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21573–21612, 2023.

Ying, H., Wu, Z., Geng, Y., Wang, J., Lin, D., and Chen, K. 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.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., YuYue, Dai, W., Fan, T., Liu, G., Liu, J., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, R., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Wu, Y., and Wang, M. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

Zheng, K., Han, J. M., and Polu, S. 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 7 summarizes the data domains used in the main experiments, and Table 8 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 7. 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 8. 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 9 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 9. 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 summarizes the aggregate manual audit of the automatic proof-level semantic judge. This appendix records the full audit table, 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 11 shows one representative positive audit case.

Table 10. Manual audit of the LLM-based proof-level semantic evaluation. The upper panel audits judge-accepted candidates; the lower panel audits a separate diagnostic set of verifier-passing candidates that were not accepted by the judge. Percentage columns are marked with %; n and uncertain columns are counts.

Judge-accepted candidates

Setting	n	Manual accepted (%)	Manual rejected (%)	Uncertain n	FPR upper bound (%)↓
Reference-Statement	150	94.7	4.0	2	5.3
Predicted-Statement	80	98.8	1.3	0	1.3

Verifier-passing candidates not accepted by the judge: diagnostic sample

Setting	n	Manual accepted in sample (%)	Manual rejected in sample (%)	Uncertain n
Reference-Statement	40	92.5	5.0	1
Predicted-Statement	30	90.0	10.0	0

Table 11. 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 12 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 12. 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 12, 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 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 13 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 13. 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`	$2 \to \perp$	N/A	failure

D. Checkpoint and Curriculum Details

Checkpoint selection is performed on the LeanWorkbook-ISA validation split. Table 14 reports the complete equal-budget RL ablation summarized in the main text. Table 15 reports the full validation checkpoint table, and Table 16 reports the corresponding test numbers and verifier-call budgets for the equal-budget ablation checkpoint and the final selected checkpoint. Figure 2 visualizes the verifier-feedback composition for equal-budget RL variants and the validation sweep for 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.

Table 14. Equal-budget RL ablation on LeanWorkbook-ISA. Metrics are percentages; all RL variants use 300 steps.

Method	Calls	pass@1↑ (%)	SemPass@1↑ (%)	pass@4↑ (%)	SemPass@4↑ (%)	N
Theory-SFT	0	50.6	49.8	56.1±0.9	55.4±0.8	243
GRPO-V-DO-300	7,170	51.4	51.0	57.7±0.8	57.2±0.9	243
GRPO-MVP-Uniform-300	6,948	51.0	50.6	57.9±1.3	57.4±1.3	243
GRPO-MVP-DO-300	7,428	52.7	52.3	58.9±0.9	58.1±1.1	243
GRPO-MVP-DO-λ0.3-300	7,545	51.9	51.4	58.1±1.1	57.4±0.8	243

Table 15. 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

Figure 2 restores the training-diagnostics view from the full draft. The left panel is diagnostic rather than a primary metric:

Table 16. 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

all-fail groups provide little group-relative contrast, while mixed groups provide useful advantages for GRPO. The right panel documents that the final checkpoint was fixed by validation before test evaluation.

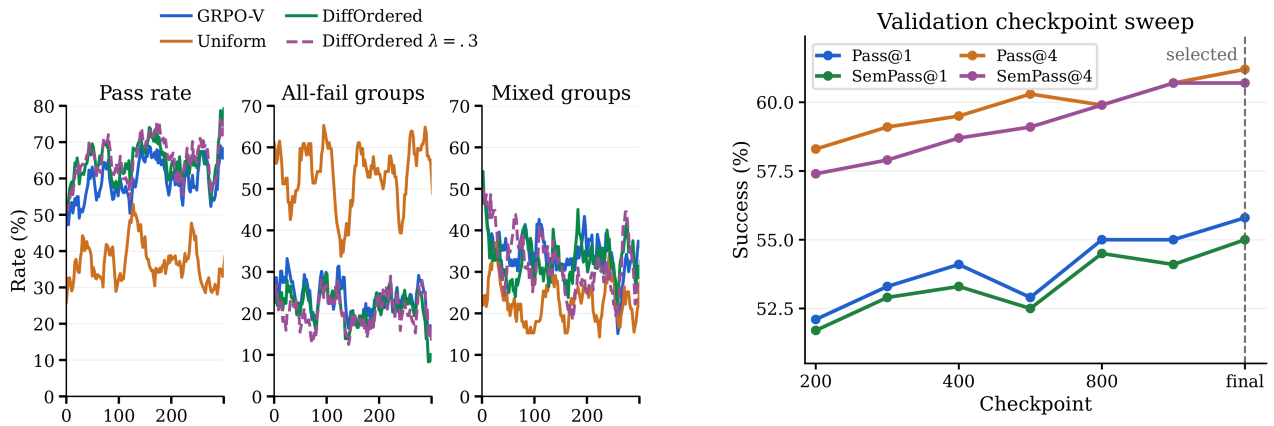

Figure 2. Training diagnostics and validation sweep. Left: verifier-feedback rates for equal-budget RL variants. Right: final validation sweep with the selected checkpoint fixed before test evaluation.

Figure 3 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 17 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, 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 frozen signal families defining the deterministic ordering are summarized below.

Table 17. 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

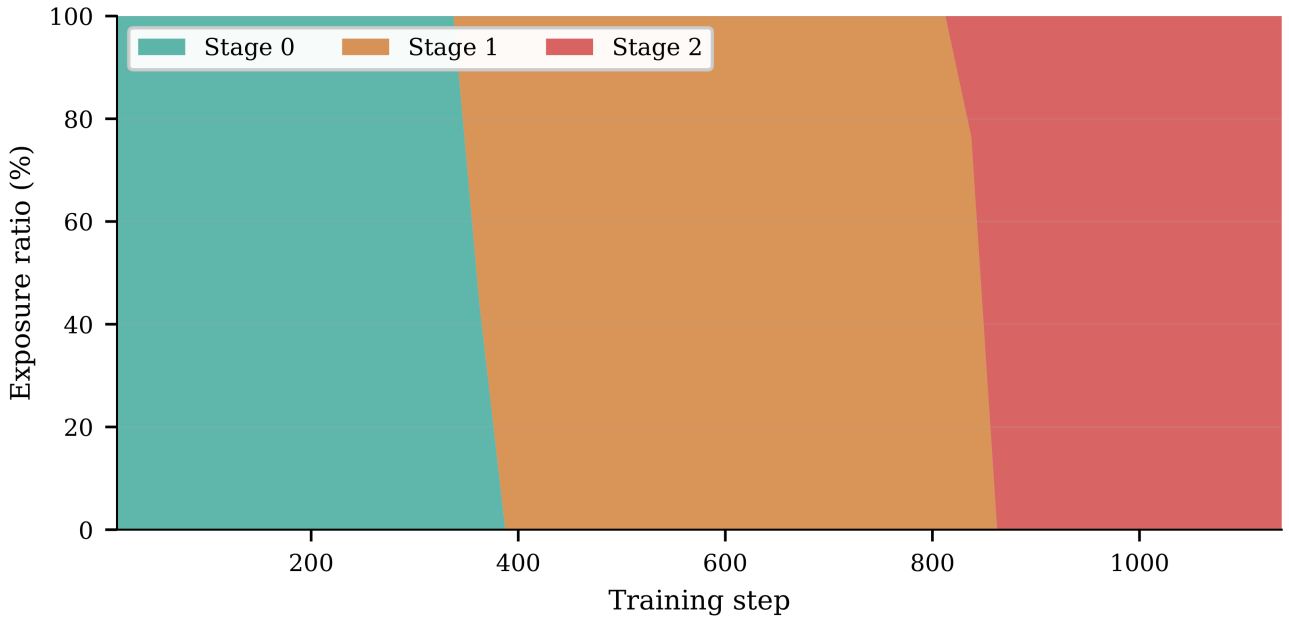

Figure 3. Curriculum exposure for the final GRPO-MVP-DifficultyOrdered run.

E. Reproducibility and Compute Disclosure

Table 18 summarizes the main training recipe. 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 18. Main model and optimization recipe. We report the settings that define the experimental recipe; lower-level optimizer defaults follow the training stack defaults unless specified here.

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 19 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 19. 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 20 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.

Table 20. Teacher construction statistics. Raw, Compile success, and Semantic accept count pre-filter candidates; Kept counts retained 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

For the Statement Teacher, the *Semantic accept* column is a pre-filter candidate count, whereas *Kept* is the example-level retained count. The gap therefore combines candidate-to-example canonicalization, source-side duplicate removal,

miniF2F-test overlap removal, and final construction filters; Table 9 reports only the explicit evaluation-overlap checks, not an itemized ledger for every pre-filter 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.

Table 21. 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 22. 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

RAG case study. Table 22 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 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 6. Table 23 gives the full pass and SemPass version for both GRPO-MVP-DifficultyOrdered and Theory-SFT, and Table 24 summarizes coarse failure buckets for the final system.

Table 23. 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 24 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 descriptive and use the evaluation set as denominator.

Table 24. 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

H. Statement-Domain Diagnostics

The lower Herald-ISA statement-translation score in Table 3 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 on theorem syntax, 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, judge, and retrieval-summary prompts. The full prompt files are provided in the supplementary artifact.

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 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."
}
```

RAG chunk-summary prompt excerpt

```
The RAG chunk-summary prompt is lengthy; we include the core instruction
excerpt 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 25 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 25. 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