ASSUMPTION LAUNDERING IN PROVER-CERTIFIED PROCESS REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work supervises language-model reasoning with a theorem prover: each step is translated into first-order logic, the prover checks that the step's premises entail its conclusion, and the verdict is used as a step-level reward. When the cited premises fall short, some pipelines let the model add an assumption to close the gap. We show that this makes the certificate weaker than it looks, because the model being supervised writes its own obligation: a proof from the model's own axiom is reported as a proof of the model's step, a failure we call assumption laundering. We audit it on 15,733 steps sampled from the base model of a released pipeline on ProntoQA and ProofWriter, whose symbolic generators let us decide both whether a step is valid and whether an assumption is entailed by the problem. Where the cited premises do not entail the conclusion, 87% of accepted certificates hold only because of an assumption the model added, and the certificate cannot tell an assumption the problem entails from one it does not. A model that assumes what it needs to prove is accepted on every step, so no threshold on the verdict can repair this; the assumption has to be licensed by a reference the model does not control. Enforcing this condition raises informedness (acceptance of valid minus acceptance of invalid steps) on ProntoQA from 0.09 to 0.41, and to 0.49 after we also repair a separate compiler defect that we trace to the demonstrations in the released prompt; repairing the compiler alone leaves it at 0.09. A second pipeline that constrains assumptions differently behaves as the condition predicts. Where no reference exists, disallowing assumptions altogether does not give a usable step reward, but it does give a high-precision data filter: 13% of responses kept at 93% answer accuracy against a 72% base rate, ahead of a learned process reward model at the same yield.

1 INTRODUCTION

Process supervision requires a signal that says whether an individual reasoning step is sound. Learned process reward models (Lightman et al., 2024; Wang et al., 2024) provide such a signal cheaply but miss many step-level errors (Song et al., 2025), and their benchmark accuracy is a weak predictor of downstream utility (Wen et al., 2025). A theorem prover offers an alternative that appears immune to both: recent pipelines compile natural-language reasoning into logic, discharge the resulting obligation with an automated prover, and use the verdict, which we call a certificate, as a reward, a filter or a judge (Xu et al., 2026; Pan et al., 2023; Olausson et al., 2023; Ye et al., 2023; Feng et al., 2026).

The guarantee, however, attaches to the obligation the prover is given, and who writes that obligation differs by design. Logic-LM, LINC and SatLM formalize the whole problem once. When certification is restricted to a step's cited premises, a background rule the step leaves unstated makes a valid step unprovable, and some step-level pipelines therefore have a language model supply the missing premise: in LogicReward the policy states it as an assumption inside its step, in VeriCoT a premise may be typed as commonsense. The audit concerns this design. In the instance we audit the premise is written by the policy under evaluation, which has seen the conclusion it must justify and may add an assumption whenever the cited premises fall short. Figure 1 illustrates the consequence. Two steps of one response, three lines apart, compile into the same template, are discharged by the same prover and accepted with the same provenance; in one the model asserted the converse of a rule its corpus states, in the other it restated a rule the corpus entails. Nothing the pipeline observes

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

distinguishes them. We refer to this failure as *assumption laundering*: a proof from the model's own axiom presented as a proof of the model's step.

Prior work has established that autoformalization by a language model (Wu et al., 2022) can be unfaithful to its source (Kim et al., 2026) and that fixed verifiers can be gamed under optimization (Skalse et al., 2022; Helff et al., 2026). The failure here is neither. The obligation in Figure 1 is translated faithfully and is true as written; what it does not establish is the step, because the axiom that makes it true is the model's own. The rollouts we audit come from a policy that was never trained against this certificate, so the failure precedes any optimization and needs no intent: an assumption that closes the goal is what a model writes once its own reasoning has gone wrong. A more faithful compiler does not remove this and, as we show, makes it more frequent. We ask three questions of such a certificate once it is deployed as a training signal: how often an accepted certificate rests on an axiom the model supplied rather than on the premises it cited; what it guarantees in the worst case and under what condition; and what enforcing that condition costs and buys. The first needs a reference the model did not write. ProntoQA (Saparov & He, 2023) and ProofWriter (Tafjord et al., 2021) ship the symbolic programs that generated their questions, so on these corpora both the validity of a step and the licensing of an assumption can be decided rather than annotated, and the audit is anchored there. The instance we audit is LogicReward (Xu et al., 2026), which is released with the prompts, data and code needed to reproduce it and uses the verdict as a training reward, described as "a reliable reward signal even without ground-truth labels". The condition the audit yields is then tested on a second design that constrains assumptions differently (Feng et al., 2026), and the compiler defect against two independently authored compilers (Pan et al., 2023; Olausson et al., 2023).

Our contributions are as follows.

- We introduce a provenance audit for prover-certified process rewards: each obligation is discharged with and without the model's assumption, and the assumption is read against the corpus program. On 15,733 rollout steps, 87% of accepted certificates on locally unsupported steps are carried by the model's assumption, which on ProntoQA the program entails on 91% of correct steps and does not license on 82% of incorrect ones (§5.2).
- We show that unconstrained bridging admits a strategy that is accepted on every step, which no threshold on the verdict can detect, and we evaluate licensing by a reference the model does not control as the way to block that degeneracy; whether it restores discrimination in practice is an empirical question (§4).
- We enforce the condition in two forms. Admitting only assumptions the reference licenses raises informedness from 0.09 to 0.41 on ProntoQA and from 0.15 to 0.22 on ProofWriter, which places the failure in the provenance of the assumption rather than in the prover or the threshold; this form needs a reference and serves as a diagnostic. Withdrawing assumptions altogether needs none and is the form we recommend: as a filter it keeps 13% of responses at 93% answer accuracy against a 72% base rate, ahead of a learned process reward model at matched yield, though not separable there from a length baseline (§5.3).
- We show that repairing the compiler is not a substitute. A defect that renders proper names as predicates accounts for most of the invalid steps a constructed audit accepts and is attributable to the demonstrations in the released prompt; repairing it leaves the rollout reward at 0.09 on ProntoQA, raises ProofWriter's from 0.15 to 0.25, and on both corpora *increases* the share of certificates the assumption carries (§6).

2 Related Work

Faithfulness of autoformalization. Formal verification guarantees proof validity, not formalization faithfulness. Kim et al. (2026) term the resulting gap formalization gaming and observe axiom fabrication and premise mistranslation in a two-stage Lean 4 pipeline, judged against a second formalization. Faithfulness is usually measured against a reference, by alignment scoring (Lu et al., 2025) or bidirectional equivalence (Poiroux et al., 2025); roundtrip verification (Amrollahi et al., 2026) and structure-preserving decomposition (Cabral et al., 2026) have been proposed as repairs; and error taxonomies give the misrepresentation of an object's type a category of its own (Wang et al., 2026). All of this asks whether the compiled formulas mean what the text says. The failure

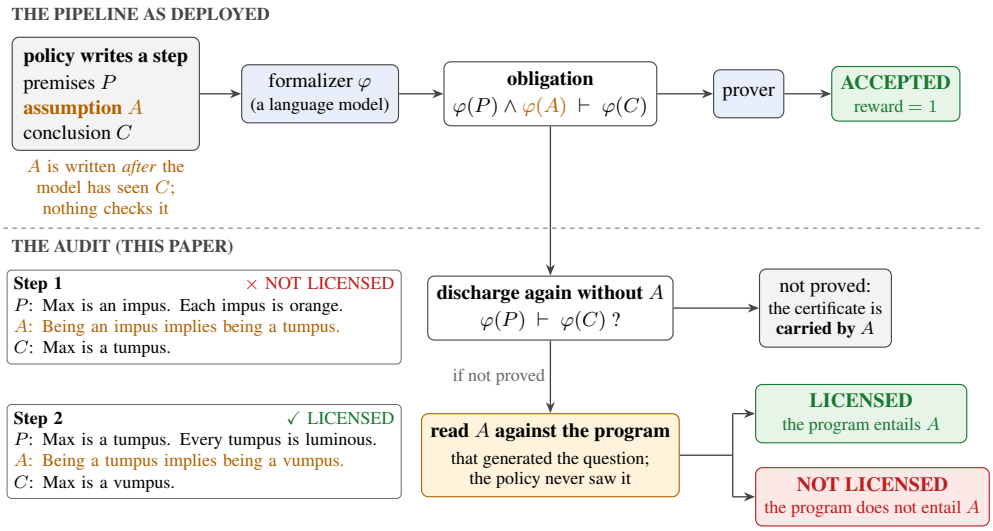

Figure 1: The pipeline and the audit. Top: the policy writes a step (P, A, C), a formalizer compiles it and a prover discharges $\phi(P) \wedge \phi(A) \vdash \phi(C)$; nothing upstream checks A. Bottom: the audit discharges the obligation again without A, which identifies the certificates the assumption carries, and reads A against the corpus program that generated the question. Left: two steps of one ProntoQA rollout, three lines apart, with the same verdict on the top path; the program entails the second assumption and not the first (it states the converse), and, decided on the same program, the second step is valid and the first is not.

studied here survives a faithful translation: the obligation is true, and what it proves is the axiom the model added.

Verifier gaming and reward benchmarks. Helff et al. (2026) show that models trained by reinforcement learning with verifiable rewards (RLVR) on inductive tasks produce instance-level enumerations that satisfy an extensional verifier, that the behaviour is absent in non-RLVR models, and that an isomorphism-invariant verifier removes the incentive; related work builds testbeds for such gaming (Khalifa et al., 2026). Optimizing against a learned reward degrades true performance (Gao et al., 2023), and reward-model benchmark accuracy is a weak predictor of downstream utility (Wen et al., 2025; Malik et al., 2026). In these settings the verifier is fixed and the failure is learned under optimization against it. In the setting studied here the obligation is written by the model, and the failure is present in policies that never saw the certificate, including one never trained with RLVR (Appendix B.3).

3 Preliminaries

Prover-certified process rewards. We reproduce LogicReward (Xu et al., 2026) from its original prompts and data. It follows the symbolic chain-of-thought line of Xu et al. (2024; 2025) but discharges each step with a prover; Logic-LM (Pan et al., 2023) and LINC (Olausson et al., 2023), two independently authored compilers, serve as cross-implementation controls in §6. A reasoning step, written by the model under evaluation (the *candidate*), is a triple (P, A, C) of cited premises, stated assumption and drawn conclusion. A formalizer emits a first-order theory $\phi(P)$, $\phi(A)$, $\phi(C)$, and a prover decides $\phi(P) \wedge \phi(A) \vdash \phi(C)$; the step is accepted if the obligation is discharged. The released pipeline discharges obligations in Isabelle/HOL through Sledgehammer, which passes first-order goals to external provers; we transcribe the same obligations to TPTP and call two of those provers, Vampire (Kovács & Voronkov, 2013) and E (Schulz et al., 2019), directly. The released code path, with only this prover call replaced, reproduces every finding below within the confidence intervals of our reimplementation (Appendix A.1).

Informedness. With $y \in \{\text{valid}, \text{invalid}\}$ the gold validity of a step, we report acceptance on valid steps A^+, acceptance on invalid steps A^-, and *informedness* $\text{Inf} = A^+ - A^-$ (Youden's J), which is 1 for a perfect certifier, 0 for any verdict independent of y, and insensitive to class imbalance and to certifiers that accept everything.

Reference semantics. ProntoQA and ProofWriter ship the symbolic programs that generated their questions, so two properties of a step can be decided rather than annotated, both discharged by Vampire. A step is *conclusion-valid* when its conclusion is a first-order consequence of the question's program, and *locally supported* when the premises it cites entail its conclusion by themselves. Informedness is measured against conclusion-validity, the property the certificate claims to establish. The two come apart in both directions: a step can be conclusion-valid without being locally supported, and a locally supported step can carry a false conclusion when an earlier step was wrong. The second case is the inherited-error channel of §5.2 and is a ceiling on what any step-local certificate can reach, not a failure of the certificate. An assumption is *licensed* when the question's program first-order entails it, decided by the same prover and the same compilation of the program used for conclusion-validity. It need not appear verbatim among the program's rules, and a rule reachable only through a contrapositive counts as licensed. Appendix A.2 reports the stricter readings, under which an assumption counts only if forward chaining derives it or if it appears in the program verbatim; those give a higher unlicensed share, and the difference falls almost entirely on steps whose conclusions are correct.

Label semantics. The corpora's own forward-chaining closure, which any evaluation reusing their labels inherits, is not equivalent to first-order consequence, because a closure cannot derive contrapositives: "Wren is bright; yumpuses are not bright; so Wren is not a yumpus" lies outside it, is used by the model, and is rightly accepted by the prover under audit. The difference is substantial, 45% of ProntoQA's and 7% of ProofWriter's closure-invalid rollout steps being first-order valid with none moving the other way, and it shifts the constrained certificate of §5.3 from 0.21 to 0.34 on ProntoQA. We use first-order labels throughout and recompute every headline quantity under both semantics in Appendix B.1.

Data. The 15,733 audited steps are the policy's own rollouts: Qwen3-8B (Yang et al., 2025), the pipeline's own base model, which also serves as the formalizer, sampled at temperature 1.3 with eight samples on 300 held-out ProntoQA questions and 150 ProofWriter questions, each response split into (P, A, C) triples. A constructed audit of 3,037 steps, built by local counterfactual perturbation of gold chains, pairs each valid step with invalid siblings over the same rule and individual (affirming the consequent, denying the antecedent, or substituting another individual from the context); it is used in §5.3 to quantify how far a constructed benchmark overstates the reward and in §6 to isolate the second defect. One hundred development groups are frozen and excluded from every held-out figure. The audited pool is 10,748 ProntoQA steps and 4,985 ProofWriter steps; the oracle maps 4,985 of ProofWriter's 6,667 extracted steps onto its program, the remainder being mostly relations between two named individuals its reader cannot ground. A step the oracle cannot map is excluded from step-level figures but not from its response, and a response is dropped only when none of its steps can be mapped, which keeps 1,064 of 1,200 ProofWriter and 2,382 of 2,400 ProntoQA responses in the filter of §5.3; the "every step certified" criterion is applied over all of a response's steps, and the step-validity column is computed over the mapped ones. Rollouts from one question are not independent, and bootstrap intervals over responses or steps resample questions. Table 3 lists every data pool in the paper and the tables that use it.

4 Provenance Audit

4.1 The licensing condition

The following observation identifies a worst-case degeneracy of a certificate of this form.

Observation 1 (Unconstrained bridging has no floor)**.** *Let a certifier accept a step (P, A, C) iff $\phi(P) \wedge \phi(A) \vdash \phi(C)$ for a compiler ϕ and a sound prover $\vdash$, where the candidate chooses A after seeing C from an admissible set $\mathcal{A}$ that contains the conclusion of every step. Then there is a candidate strategy, namely assuming the conclusion, that is accepted on every step; against such a candidate the verdict is independent of the step's validity and* $\text{Inf} = 0$.

The proof is the substitution $A := C$. The observation describes a worst case rather than how models behave: it says the certificate has no guaranteed floor, not that its verdict is uninformative for a particular policy, and how much information it carries in practice is the empirical question §5 answers. What it does rule out is a repair by thresholding. Given $(\phi(P), \phi(A), \phi(C))$ and a verdict, a true background fact and an axiom introduced because it closes the goal are the same object, so no threshold on this certificate separates them; a fix has to change what may be assumed. Restricting $\mathcal{A}$ to sentences licensed by a reference the candidate does not control removes the degeneracy but does not by itself make the certificate informative, since the compiler may be unfaithful and the cited premises may be wrong. The same structure arises whenever an accepted set contains an element the candidate can always produce, as when an extensional verifier accepts an enumeration in place of a rule (Helff et al., 2026); the step-level case is the extreme one, because that element is the goal itself.

4.2 Discharging the obligation with and without the assumption

For each accepted step the compiler under test discharges three obligations instead of one: (1) premises together with the candidate's assumption, which is the released rule; (2) premises alone; (3) the assumption alone. A certificate is *carried by the assumption* when (1) closes and (2) does not, i.e. the certificate establishes the conclusion conditional on the candidate's assumption rather than from the cited premises alone. Two further readings require neither model nor prover and are available only where a corpus ships its program: (4) whether the cited premises entail the conclusion, decided on the program; (5) whether the carrying assumption is *licensed*, that is, first-order entailed by the program, after a deterministic reader has mapped the English sentence onto a formula (Appendix A.2). Readings (1)–(3) apply to any pipeline that permits assumptions; readings (4)–(5) are the reason the audit is anchored on ProntoQA and ProofWriter.

4.3 Two ways to enforce the condition

The condition names a reference but does not say what to do when none is available, and the two rules we evaluate are the ends of that choice. The *licensed* rule accepts a step when the obligation closes and, where the assumption was needed to close it, the program entails that assumption: the condition applied as stated, with the corpus program as the reference. The *premises-only* rule accepts a step only when the cited premises close the goal unaided, the special case $\mathcal{A} = \emptyset$, which needs no reference and can be run anywhere. Neither is a formalizer fix; both change what the certificate is permitted to certify, the first at the cost of requiring a reference, the second at the cost of every step whose justification genuinely rests on an unstated premise. The two are not interchangeable. On a corpus that ships a program, anything the licensed rule can decide about an assumption the program can also decide about the conclusion, so we report the licensed rule as a diagnostic: it isolates how much of the failure is attributable to the provenance of the assumption, holding the prover, the compiler and the threshold fixed. It is also the form the condition takes wherever a reference constrains what may be assumed without settling what is to be proved, as when the reference is a document set or a verified premise library rather than the generator of the question. Figure 4 in Appendix C.2 pictures the three admissible sets. Premises-only is what we recommend for deployment, and the comparisons against the learned process reward model and the length baseline below use it, since those baselines receive no reference either.

5 Experiments

5.1 Setup

Unless stated otherwise, experiments use the rollouts and constructed audit of §3 (Table 3). We compare LogicReward with its original prompt (Orig), LogicReward with the formalizer repair of §6 (+Demo: three worked examples with a named individual added to the formalizer's prompt), and the licensed and premises-only rules under each prompt, against a learned process reward model (Qwen2.5-Math-PRM-7B, Zhang et al., 2025) and a length baseline that prefers the shorter response. Informedness and ranking accuracy are reported with bootstrap 95% confidence intervals over questions; shares are reported with Wilson intervals.

Table 1: Provenance of certificates on rollout steps, by gold label and whether the cited premises alone entail the conclusion (decided on the corpus program). Accept.: acceptance rate; Carried: share of accepted certificates provable with the assumption but not without it. Shaded rows: locally unsupported steps. Wilson 95% CIs (under +DEMO, shaded carried shares only).

				ORIG		+DEMO	
Corpus	Gold	Prem. entail	#	Accept.	Carried	Accept.	Carried
ProntoQA	valid	yes	8,575	0.891 [.884,.897]	0.016 [.013,.019]	0.988	0.001
ProntoQA	valid	no	1,952	0.605 [.583,.626]	**0.952** [.938,.963]	0.687	**0.991** [.984,.995]
ProntoQA	invalid	yes	65	0.923 [.832,.967]	0.000 [.000,.060]	0.985	0.000
ProntoQA	invalid	no	156	0.679 [.603,.748]	**0.774** [.685,.843]	0.782	**0.828** [.751,.885]
ProofWriter	valid	yes	2,475	0.531 [.511,.551]	0.156 [.137,.177]	0.725	0.045
ProofWriter	valid	no	1,555	0.363 [.340,.388]	**0.812** [.778,.842]	0.468	**0.922** [.900,.939]
ProofWriter	invalid	yes	121	0.579 [.489,.663]	0.157 [.090,.260]	0.694	0.048
ProofWriter	invalid	no	834	0.278 [.249,.310]	**0.612** [.548,.672]	0.330	**0.760** [.706,.807]

5.2 PROVENANCE OF ACCEPTED CERTIFICATES

Assumption-carried certificates. Table 1 crosses the gold validity label with reading (4). Where the cited premises entail the conclusion, the assumption is rarely needed and the carried share is near zero. Where they do not, 95% of ProntoQA's certificates on correct steps and 77% on incorrect steps are carried by the assumption; ProofWriter gives 81% and 61%. Because the row labels are decided on the corpus program, they do not depend on the compiler under test. The formalizer repair of §6 does not reduce this share; it increases it on incorrect unsupported steps, where a more faithful compilation lets the model's axiom close the goal more often. Alignment of the extracted triples to the source text is verified rather than assumed: on 90.2% of steps every cited premise is a verbatim context sentence or a verbatim earlier conclusion (Appendix B.3).

False-certificate channels. Not every carried certificate is laundered. On ProntoQA, 166 of the 221 first-order-invalid rollout steps receive a certificate, and reading each against the program sorts them into four channels. In 40% the certificate is carried by an assumption the program does not license, with a further 4% carried by assumptions the reader cannot parse. In 36% the cited premises do entail the conclusion and an earlier step was wrong: an inherited error that no step-local certificate can detect, and which bounds what any such reward can achieve. In 14% the compiled premises prove a non-consequence, a mistranslation. In the remaining 5% the assumption is licensed and the step is wrong for another reason. The unlicensed assumption is the largest channel on ProntoQA, the second largest on ProofWriter (Appendix B.1), and the only one a stricter threshold cannot affect.

Licensed and unlicensed assumptions. On correct steps the carrying assumption is, in 91% of cases, entailed by the question's own program but restated as a conditional rather than cited, a citation failure rather than a soundness failure. On incorrect steps it is, in 82% of cases, a rule the program does not entail (Table 13; ProofWriter 73% and 54%). Figure 1 shows one such pair from a single response: the same obligation shape, the same verdict and the same provenance, distinguished only by the corpus program.

5.3 THE CERTIFICATE AS A REWARD AND AS A FILTER

Step-level reward. Figure 2 scores the certificate on the policy's own steps. The original certificate reaches informedness 0.64 on the constructed audit and 0.09 on rollouts. Part of that gap is inherited error, which caps informedness near 0.7 on ProntoQA, since 65 of its 221 invalid steps are entailed by their cited premises (Table 1); the rest is laundering and mistranslation. It also does not outperform the learned critic it was introduced to replace: the learned process reward model reaches 0.23 at its natural threshold and 0.36 with a threshold tuned on the gold labels, an advantage the certificate is not given.

A faithful compiler is not a sufficient repair. Repairing the formalizer changes the reward on one corpus and not the other: on ProntoQA informedness stays at 0.09, on ProofWriter it rises from 0.15

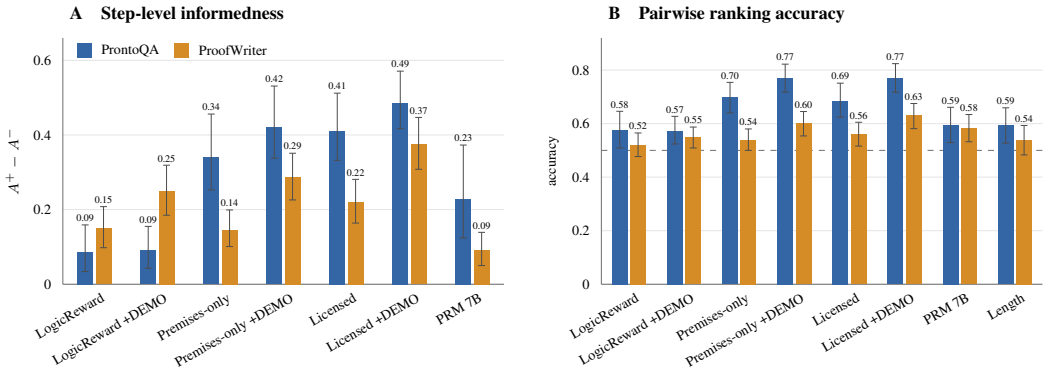

Figure 2: The certificate as a step-level reward on the policy's own rollouts. A: informedness under first-order labels. B: pairwise ranking accuracy of whole responses against the gold process score (671 ProntoQA and 1,557 ProofWriter pairs); dashed line, chance. 95% bootstrap intervals resampling questions. +DEMO: the repaired formalizer of §6; PRM: Qwen2.5-Math-PRM-7B at its natural threshold; Licensed uses the corpus program and is a diagnostic (§4.3). Values in Table 10.

to 0.25. Neither reaches what the constrained rules attain, and on both corpora the repair *increases* the share of certificates carried by the assumption (Table 1), because a more faithful compilation lets the model's axiom close the goal more often.

Enforcing the condition. Accepting only assumptions the program entails raises informedness from 0.09 to 0.41 on ProntoQA and to 0.49 with the formalizer repaired; on ProofWriter, from 0.15 to 0.22 and from 0.25 to 0.37. Withdrawing assumptions altogether is the more restrictive rule and the less informative one: premises-only reaches 0.34 and 0.42 on ProntoQA and 0.15 and 0.29 on ProofWriter, below the licensed rule in every cell. It is also the more expensive rule. Of the conclusion-valid ProntoQA steps the original certificate accepts, premises-only rejects 1,246; the licensed rule rejects 122, recovering 90% of the coverage premises-only gives up (60–67% on ProofWriter).

Ranking. Both constrained rules improve the ordering of whole responses by process quality: pairwise ranking accuracy against the gold process score rises from 0.58 to 0.77 on ProntoQA and from 0.52 to 0.63 on ProofWriter, against length baselines of 0.59 and 0.54. The improvement does not carry over to answer selection. Used as a best-of-8 selector, no reference-free rule exceeds random selection by more than 0.02 on any of six corpora (Appendix C.4); the licensed rule cannot be evaluated there, since four of the six ship no program. The constrained certificate orders responses by the soundness of their steps, not by whether they end in the right answer.

Filter. A rule too weak to score every step can still select a subset, which is what data curation by a verifier requires (Zelikman et al., 2022). Applied at its own threshold, accepting a response only if every step is certified, premises-only retains 13% of ProofWriter responses at 0.93 answer accuracy and 0.99 conclusion-valid steps, against a base rate of 0.72 and 0.82 (Figure 3); at the same yield the learned process reward model reaches 0.80 and the length baseline 0.83. Against the PRM the paired gap is $+0.133$ [.054,.237]; against the length baseline it is $+0.098$ [$-.006$,.231], which does not exclude zero, so on this corpus we can separate the certificate from a learned critic but not from preferring shorter answers. The licensed rule reaches 0.94 at a yield of 18% and separates from both, but it consumes the corpus program and so is not comparable with baselines that receive no reference. ProntoQA has no headroom in answer accuracy at a base rate of 0.98, though step validity still rises from 0.983 to 0.996. Appendix C.4 measures what premises-only buys on four corpora that ship no program. Whether the selection improves a trained policy is not established: a matched LoRA-DPO comparison (Hu et al., 2021; Rafailov et al., 2023) across six selection rules does not move the policy under any of them, the gold process score included, and so cannot rank them (Appendix D.2).

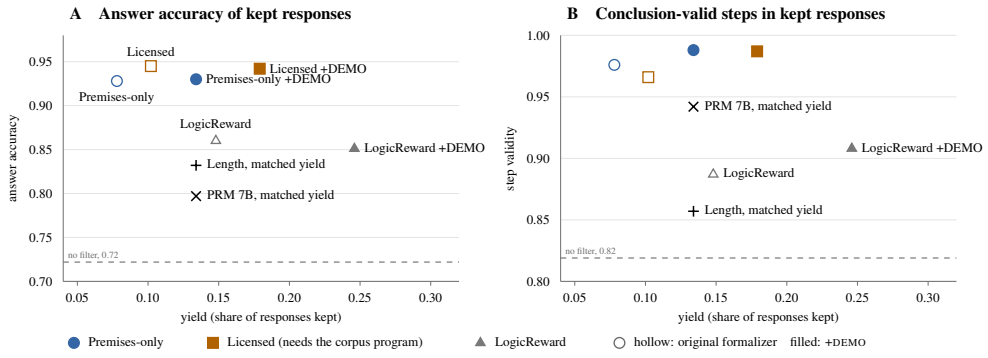

Figure 3: The certificate as a filter on ProofWriter responses ($n = 1{,}064$): keep a response only if every step is certified, so each rule sits at its own threshold. Shape: rule; filled: repaired formalizer. The PRM and the length baseline have no such threshold and are taken at the yield of Premises-only +DEMO; Licensed requires the corpus program. Values, paired gaps and ProntoQA in Table 11.

Sensitivity to the licensing relation and the reader. Deciding licensing by first-order entailment rather than by the stricter readings of Appendix A.2 (the corpora's forward chaining, or verbatim membership in the program) changes figures on correct steps, where the additionally licensed assumptions are contrapositives, and on ProofWriter raises the unlicensed share on incorrect steps from 0.49 to 0.54; on ProntoQA every figure on incorrect steps is unchanged. The licensed rule also inherits the reader's coverage, since an assumption the reader cannot parse is not accepted; admitting those instead moves informedness by at most 0.03 in every cell of Table 10 and the filter yields by at most 0.04 (Appendix A.2).

Other corpora. Readings (1)–(3) compare the pipeline with itself, so the carried share can be measured where no reference exists. On FOLIO (Han et al., 2024), ProverQA (Qi et al., 2025), Multi-LogiEval (Patel et al., 2024) and e-SNLI (Camburu et al., 2018) the certificate accepts far fewer steps (0.11–0.28, against 0.82 on ProntoQA), and between two fifths and three fifths of what it accepts is carried by the model's assumption (Table 6). Refusing more does not make the surviving certificates safer; it concentrates them on the steps the model had to bridge. Two reference-free checks bound the effect of inertly compiled premises at 0.014 (Appendix B.2); a translation that appears faithful but is not cannot be excluded without a reference, so on these corpora the carried share is an upper bound on laundering. Substituting a judge model for the corpus reader recovers a usable ordering of assumptions but not a usable rate, the estimated unlicensed share moving sixfold with the judge configuration (Appendix C.5).

A second pipeline. The condition makes a prediction about any pipeline that constrains what may be assumed. VeriCoT-style verification (Feng et al., 2026) types every premise and refuses a step whose premises are not grounded in the context, constraining assumptions at the premise level rather than at the goal. Reimplemented and run on the same steps and labels, it behaves as predicted in both directions: on ProntoQA its informedness is 0.26 against 0.09, with only 3.5% of accepted certificates using a model-written premise, while on ProofWriter it refuses 85% of steps and its informedness falls below that of the original certificate (Appendix D.1). Constraining assumptions buys resistance to laundering and costs coverage, and the same trade-off appears in a pipeline designed independently of ours.

Policy, temperature and formalizer. The failure is not an artifact of sampling temperature, of a weak policy or of the formalizer. At temperature 0.7 both the carried share and the unlicensed share increase, and a $4\times$ larger policy is worse still: 94% of its first-order-valid steps are locally unsupported and its informedness falls to 0.016, indistinguishable from zero. The carried share also replicates in a policy never trained with RLVR, and replacing the formalizer by a $4\times$ larger one raises the carried share further while the constrained rules keep their lead, at 0.42–0.48 against 0.12 on ProntoQA and 0.30–0.40 against 0.26 on ProofWriter (Appendix B.3). The pools differ in several

Table 2: Constructed audit (3,037 held-out steps): acceptance on valid (A^+) and invalid (A^-) steps and informedness, for the released prompt, the three-demonstration repair and the deterministic constant contract. FOLIO and e-SNLI in Table 14.

	ORIG			+DEMO			CONTRACT		
Corpus	A^+	A^-	Inf	A^+	A^-	Inf	A^+	A^-	Inf
ProntoQA	0.891	0.250	0.642	**1.000**	**0.000**	**1.000**	0.989	0.006	0.983
ProofWriter	0.724	0.181	0.544	**0.880**	**0.000**	**0.880**	0.849	0.019	0.830

respects at once, so they show that neither capability nor temperature removes the failure without isolating what produces it; the one causal claim we make is the demonstration-set attribution of §6.

6 A SECOND DEFECT: PROPER NAMES COMPILED AS PREDICATES

Observation 1 assumes a sound prover, not a faithful compiler. The compiler is separately defective, and the two failures are independent: repairing this one leaves laundering untouched, and enforcing the licensing rule does nothing for this one.

Defect. On the constructed audit the original certifier accepts a quarter of the invalid steps on ProntoQA and a fifth on ProofWriter (Table 2). Splitting acceptance by how the invalid sibling was built shows that the failure is not logical: the classical fallacies are caught, affirming the consequent at 0.000, while a step that applies the right rule to the wrong individual is accepted three times in four on ProntoQA and half the time on ProofWriter (Table 15, Figure 5). The cause is visible before any prover runs. In the compiler's own logical forms, 12.7% of proper-name occurrences on ProntoQA and 30.1% on ProofWriter are rendered as predicates rather than constants (Table 16); a statement about the kind `max` is strictly weaker than one about the individual Max, and any rule mentioning that predicate discharges the obligation built from it. A deterministic rewrite of those tokens applied to the released output, with no model call (the constant contract, CONTRACT), recovers informedness 0.98 and 0.83, consistent with a surface defect rather than a reasoning failure.

Attribution. That autoformalizers confuse constants and predicates is known (§2); what causes a particular compiler to do so is not. The original prompt contains eight demonstrations, all with kind subjects ("Bears are quiet") and none with a named individual. We hypothesize that the compiler renders a name as a kind because its demonstrations never show a named individual. Three named-subject demonstrations raise informedness on both corpora to 0.88–1.00, whereas count- and form-matched demonstrations with kind subjects change nothing (Table 18). The attribution rests on injection: Logic-LM's ProofWriter prompt keeps every named individual as a constant, and rewriting its single demonstration so that facts about individuals become facts about kinds ("Anne is quiet" $\rightarrow$ "Bears are quiet"), with nothing else changed, raises its predicate rate from 0.000 to 0.609 on the same sentences. The repair produces a correct obligation rather than an easier one, bidirectional equivalence to the corpora's ground truth rising from 0.55 to 0.84 and from 0.44 to 0.83 (Table 17); prover backend, formalizer scale and model family do not change the picture (Appendix C.2).

7 CONCLUSION

When the model writes the obligation and may add axioms to it, the prover's guarantee covers the inference from the axiom and says nothing about the axiom. On the policy's own rollouts most certificates on locally unsupported steps rest on such an axiom, and the certificate is nearly uninformative where a constructed benchmark of the same pipeline looks strong. A more faithful compiler does not repair this; requiring the closing axioms to be licensed by a reference the model does not control does. The rule is only as portable as the reference. Without one, withdrawing assumptions still yields a high-precision data filter, but whether data selected this way improves a trained policy remains open (Appendix D.2), and on natural-language corpora the pipeline refuses most valid steps, so there the carried share bounds laundering rather than measuring it (Appendix C.3).

AI USE STATEMENT

In this work, we used generative AI tools for research execution and to aid and polish the writing: under author supervision they were used to implement methods and analysis scripts, to give feedback on the design of experiments, to clean and reformat data, to help interpret results, and to improve the clarity and grammar of text written by the authors. We have not used generative AI tools to formulate the research question or the licensing condition, to formulate or prove the mathematical claim (Observation 1), to generate any dataset (the constructed audit is produced by a deterministic program from the corpora's published programs), to label data or judge the validity of any step, or to find related work. Translation and qualitative data analysis are not applicable to this work. We have reviewed all AI-assisted work: AI-written code was verified by running it against held-out checks (the oracle reproduces the corpora's published answers, frozen development groups are excluded from every held-out figure, and the released code path of the audited pipeline reproduces our reimplementation's numbers within confidence intervals), every step label in this paper is produced by an executable oracle or by the corpus' published program, and every claim in the text was checked by the authors against the tables it cites. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects and no personal data. All corpora (ProntoQA, ProofWriter, FOLIO, e-SNLI, ProverQA, Multi-LogiEval) and all pipelines (LogicReward, Logic-LM, LINC) are publicly released research artifacts used under their licences; the released LogicReward prompts are quoted only to the extent needed to state the defect. The finding that a prover-backed certificate can be carried by an axiom the model wrote is a reliability result about a class of reward signals, and it bears on safety agendas resting on auditable proof certificates produced with a learned formalizer in the loop (Dalrymple et al., 2024): a certificate is only as strong as the provenance of its obligation. We see no path to misuse; the risk this paper describes is a reliability risk, and it is larger where such certificates are trusted without an audit, which is why we release the check.

REPRODUCIBILITY STATEMENT

Prompts, the constant contract, the generator of the constructed audit, the oracle programs, the sentence-split rollout steps, and every run identifier behind each table are released with the code. Development and held-out groups are frozen by id. Every rollout number can be recomputed without a GPU from the original certificate files; the only model calls are the formalizations themselves, which are cached and released.

REFERENCES

Daneshvar Amrollahi, Jerry Lopez, and Clark Barrett. Faithful autoformalization via roundtrip verification and repair. *arXiv preprint arXiv:2604.25031*, 2026.

Rafael Cabral, Tuan Manh Do, Xuejun Yu, Wai Ming Tai, Zijin Feng, and Xin Shen. ProofFlow: A dependency graph approach to faithful proof autoformalization. In *International Conference on Learning Representations*, volume 2026, pp. 58459–58483, 2026.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-SNLI: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette Wing, and Joshua Tenenbaum. Towards guaranteed safe AI: A framework for ensuring robust and reliable AI systems. *arXiv preprint arXiv:2405.06624*, 2024.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. VeriCoT: Neuro-symbolic chain-of-thought validation via

logical consistency checks. In *International Conference on Learning Representations*, volume 2026, pp. 133170–133208, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, 2024.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. *arXiv preprint arXiv:2604.15149*, 2026. LLM Reasoning Workshop at ICLR 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Muhammad Khalifa, Zohaib Khan, Omer Tafveez, Hao Peng, and Lu Wang. Countdown-Code: A testbed for studying the emergence and generalization of reward hacking in RLVR. *arXiv preprint arXiv:2603.07084*, 2026.

Kyuhee Kim, Auguste Poiroux, and Antoine Bosselut. Do LLMs game formalization? Evaluating faithfulness in logical reasoning. *arXiv preprint arXiv:2604.19459*, 2026.

Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *International Conference on Computer Aided Verification*, pp. 1–35. Springer, 2013.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jianqiao Lu, Yingjia Wan, Yinya Huang, Jing Xiong, Zhengying Liu, and Zhijiang Guo. FormalAlign: Automated alignment evaluation for autoformalization. In *International Conference on Learning Representations*, volume 2025, pp. 102122–102149, 2025.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. In *International Conference on Learning Representations*, volume 2026, pp. 144839–144866, 2026.

Theo X. Olausson, Alex Gu, Benjamin Lipkin, Cedegao E. Zhang, Armando Solar-Lezama, Joshua B. Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Nisarg Patel, Mohith Kulkarni, Mihir Parmar, Aashna Budhiraja, Mutsumi Nakamura, Neeraj Varshney, and Chitta Baral. Multi-LogiEval: Towards evaluating multi-step logical reasoning ability of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20856–20879, 2024.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Reliable evaluation and benchmarks for statement autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17958–17980, 2025.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reasoning evaluation. In *International Conference on Learning Representations*, volume 2025, pp. 54186–54205, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Stephan Schulz, Simon Cruanes, and Petar Vukmirović. Faster, higher, stronger: E 2.3. In *International Conference on Automated Deduction*, pp. 495–507. Springer, 2019.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9460–9471, 2022.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25299–25346, 2025.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Haocheng Wang, Baiyu Huang, Yingjia Wan, Xiao Zhu, Xiaoyang Liu, Yinya Huang, and Zhijiang Guo. FormalRx: Rectify and eXamine semantic failures in autoformalization. *arXiv preprint arXiv:2607.04655*, 2026.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Xueru Wen, Jie Lou, Yaojie Lu, Hongyu Lin, Xing Yu, Xinyu Lu, Ben He, Xianpei Han, Debing Zhang, and Le Sun. Rethinking reward model evaluation: Are we barking up the wrong tree? In *International Conference on Learning Representations*, volume 2025, pp. 58769–58790, 2025.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32353–32368, 2022.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, 2024.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3052–3075, 2025.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. LogicReward: Incentivizing LLM reasoning via step-wise logical supervision. In *International Conference on Learning Representations*, volume 2026, pp. 43490–43511, 2026.

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

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488, 2022.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

A Experimental Details

Table 3: Every data pool in the paper. Steps: extracted steps the first-order oracle maps onto the question's program, of all extracted; responses: those with at least one mapped step. The formalizer is Qwen3-8B except in the formalizer control.

Pool	Policy, temperature	Size	Step labels	Used in
Rollouts, ProntoQA	Qwen3-8B, 1.3	300 questions × 8; 2,382 of 2,400 responses; 10,748 of 10,970 steps	program, first-order	Tables 1, 4, 5, 10, 11, 13, 21; Figures 1, 2, 3, 4
Rollouts, ProofWriter	Qwen3-8B, 1.3	150 questions × 8; 1,064 of 1,200 responses; 4,985 of 6,667 steps	program, first-order	as above
Constructed audit	none: gold chains, perturbed	3,037 held-out steps; valid/invalid: ProntoQA 175/525, ProofWriter 225/675, FOLIO 225/225, e-SNLI 275/275; a further 437 FOLIO and e-SNLI steps whose conclusion the context contradicts are not scored	by construction	Tables 2, 14, 15, 16, 18; Figure 5
Program-free corpora	Qwen3-8B, 0.7	8 samples on ProverQA (293 questions), FOLIO (204), Multi-LogiEval (299), e-SNLI (300)	none; final answer only	Tables 6, 7, 19, 20
Policy controls	Qwen3-8B, 0.7; Qwen3-32B-AWQ, 0.7; Mistral-7B-Instruct, 0.7	the 300 ProntoQA questions of the main rollouts	program, first-order	Table 8
Formalizer control	as the main rollouts	the main rollouts, recertified with Qwen3-32B-AWQ as formalizer	program, first-order	Table 9
Preference pool	Qwen3-8B, 1.3	450 ProofWriter validation questions × 8, disjoint from the audit	program, first-order	Table 22

A.1 FIDELITY OF THE REPRODUCTION

LogicReward reports downstream accuracy, not certificate rates, so there is no published operating point to match; we therefore compare implementations step by step. The released code path, with the prover call replaced as in §3 and run on all 15,733 rollout steps with the same inputs as our reimplementation, reaches informedness 0.082 [0.026, 0.138] and 0.158 [0.124, 0.192] against our 0.087 [0.034, 0.159] and 0.150 [0.098, 0.208]; its certificates on locally unsupported steps are carried by the candidate's assumption at 96.3% / 81.5% (correct) and 76.1% / 60.2% (incorrect) against our 95.2% / 81.2% and 77.4% / 61.2%, and its false certificates decompose into the same channels to within two points. The released path reproduces every finding of §5, so none of them is an artifact of a weaker reimplementation. We did not run the Isabelle backend itself, so acceptance could differ where Sledgehammer's other provers, or its reconstruction of a proof inside Isabelle, disagree with Vampire and E. The oracle reproduces the corpora's published answers (300/300 and 598/600 held-out questions).

Sensitivity to premise punctuation. The released implementation formalizes each step's cited premises as a field, and whether that field is split into sentences changes its verdicts. On 1,387 steps scored under both input forms, the same released code accepts 0.848 of valid steps when the premises are given as one sentence per entry and 0.218 when they are given as a single block, at syntax-validity 0.96 and 0.99. We split premises into sentences throughout, the form that matches the pipeline's own demonstrations.

A.2 THE CORPUS READER

Table 13 asks whether a carrying axiom is an instance of one of the question's own rules, so it is only as sharp as the reader that maps an English sentence onto the program. The reader is a set of regular expressions over the corpora's controlled language; no model is called and no label is annotated.

Forms accepted. Conditionals, including one with a list of consequents ("then Alex is a zumpus, an impus, and a vumpus") and one with its own premise embedded ("if Fae is a vumpus, and every vumpus is a yumpus, then Fae is a yumpus"); "being a K implies / excludes being a P"; "all Ks share the property of being P"; "X is a member of the category of Ks"; universally quantified rules with plural subject and consequent; two rules conjoined in one sentence; and, on ProofWriter, a fact dressed as a type ("the rabbit is a thing that sees the lion") and a conjunctive body written with "both".

Schematic assumptions. A *schematic* assumption—"if Sam is a vumpus, then Sam inherits all properties of vumpuses"—is not a rule of the program but a claim about all of them, and it accounts for 41% of the assumptions no surface pattern matches. We resolve it against the step's own conclusion: the instance the prover actually needed is $K \sqsubseteq Q$ for the conclusion's predicate Q, and it is licensed iff the program derives it. This is the reading most favorable to the pipeline, since a schema that overreaches is scored on the single instance that was used, so it can only lower the unlicensed share we report.

Which relation decides licensing. We decide licensing by first-order entailment, the relation that decides conclusion-validity. Two stricter readings are available and we report what they change. Under the corpora's forward chaining, which cannot derive contrapositives, 21% of the ProntoQA assumptions we would otherwise call licensed are unreachable; those steps are almost all ones whose conclusions are correct, so the licensed share on correct steps falls from 0.910 to 0.901 while every figure on incorrect steps is unchanged. Under verbatim membership in the program's rule list, a parseable fact counts as licensed without being checked against the program, and the unlicensed share on incorrect ProofWriter steps falls from 0.54 to 0.49. First-order entailment is the only one of the three consistent with how we decide step validity, and it is also the reading most favorable to the pipeline.

Coverage and its limit. The reader returns a verdict on 96.6% and 91.5% of ProntoQA's carrying assumptions on correct and incorrect steps, and on 79.9% and 76.1% of ProofWriter's; the unread remainder is what the bounds in the last column of Table 13 span. What it cannot read is mostly not a matter of surface form: schemas that name no category ("if an entity is classified as a type, it inherits all properties of that type"), sentences about the question rather than about the world ("the

question is asking whether Sam is not shy"), and, on ProofWriter, rules whose head is a relation between two named individuals.

B Sensitivity Analyses

B.1 Label semantics

Table 5 recomputes every headline rollout quantity under the corpora's forward-chaining closure and under first-order consequence, so that a reader can see which conclusions depend on the choice of §3 and which do not. Three things are invariant: the share of certificates on *correct* locally unsupported steps that are carried by the candidate's assumption (0.927 vs 0.952 on ProntoQA, 0.806 vs 0.812 on ProofWriter), the contrast in what the carrying assumption is on incorrect steps (unlicensed 0.72–0.82 against licensed 0.10–0.17 on ProntoQA under either semantics), and the ordering of the rules, with premises-only above the original certificate under either semantics on ProntoQA and level with it on ProofWriter (0.145 against 0.150 and 0.141 against 0.129, inside the intervals). Two are not. On ProntoQA the carried share on *incorrect* unsupported steps reads 0.429 under closure labels and 0.774 under first-order labels, because 45% of the steps closure calls invalid are valid and their certificates are premise-provable, which dilutes the row; and the channel decomposition moves with it (mistranslation 0.46→0.14, inherited error 0.19→0.36, laundering 0.25→0.40), because a compiler that proves a contrapositive is scored as mistranslating under closure and as correct under first-order semantics. ProofWriter, where only 7% of labels move, is stable throughout. A step-level evaluation on these corpora that reuses the published closure labels will therefore overstate compiler mistranslation and understate both inherited error and the value of any rule that rewards contrapositive reasoning.

Table 4: The corpora's forward-chaining closure labels against first-order consequence, on the rollout steps. Left: how many steps the closure calls invalid that are first-order valid (no step moves the other way). Right: informedness of the original certificate and of the constrained rule of §5.3 under each semantics. Closure labels understate the constrained rule most, because the steps it correctly accepts are exactly the contrapositive ones the closure cannot derive. Under the repaired formalizer the same comparison reads 0.079→0.091 and 0.278→0.420 (ProntoQA), 0.222→0.250 and 0.263→0.287 (ProofWriter).

			Steps labeled invalid		Inf: closure → first-order	
Corpus	Formalizer	#Steps	Closure	First-order	LogicReward	Premises-only
ProntoQA	Orig	10,748	400	221 (−45%)	0.053 → 0.087	0.206 → 0.339
ProofWriter	Orig	4,985	1,029	955 (−7%)	0.129 → 0.150	0.141 → 0.145

Table 5: Headline rollout quantities under both label semantics, original formalizer (Orig). Under +Demo the same comparison reads, for ProntoQA, carried/acc 0.966→0.991 (correct) and 0.469→0.828 (incorrect), Inf 0.079→0.091 (released) and 0.278→0.420 (premises-only); for ProofWriter, 0.922→0.922, 0.720→0.760, 0.222→0.250 and 0.263→0.287. Informedness columns are computed on the same step set as Table 4 and Table 10.

		Carried (of accepted)		Informedness		False-certificate channels		
Corpus	Labels	Correct	Incorrect	Orig	Prem.-only	Inherited	Unlicensed	Mistransl.
ProntoQA	closure	0.927	0.429	0.053	0.206	0.19	0.25	0.46
ProntoQA	first-order	0.952	**0.774**	0.087	**0.339**	**0.36**	**0.40**	**0.14**
ProofWriter	closure	0.806	0.614	0.129	0.141	0.21	0.22	0.30
ProofWriter	first-order	0.812	0.612	0.150	0.145	0.23	0.25	0.30

B.2 INERT PREMISES AND THE CARRIED SHARE

Table 6: Acceptance and carried share on six corpora under obligations (1)–(3) alone, which need no corpus program. The step counts exceed Table 1's because the first-order oracle labels 10,748 of ProntoQA's 10,970 steps and 4,985 of ProofWriter's 6,667, and the steps its reader cannot map onto the program are kept here and dropped there. The two ProofWriter rows are therefore not on the same population; what this one adds is mostly relations between two named individuals. Wilson 95% CIs; "carried" is a share of accepted certificates, not the reader-conditioned share of Table 1. The two program corpora are the temperature-1.3 pool of §3; the four natural-language corpora are sampled at temperature 0.7 from the same policy and formalizer.

			ORIG		+DEMO	
Corpus	Reference	#Steps	Accept.	Carried	Accept.	Carried
ProntoQA	program	10,970	0.821	0.148 [.141,.156]	0.915	0.145 [.138,.152]
ProofWriter	program	6,667	0.420	0.357 [.339,.375]	0.538	0.332 [.317,.348]
FOLIO	none	6,454	0.275	**0.478** [.455,.501]	0.358	**0.501** [.480,.521]
ProverQA	none	10,846	0.183	**0.465** [.443,.487]	0.354	**0.499** [.483,.515]
Multi-LogiEval	none	7,855	0.126	**0.408** [.378,.439]	0.207	**0.617** [.593,.640]
e-SNLI	none	4,875	0.108	**0.579** [.536,.621]	0.132	**0.666** [.628,.701]

"Carried" is defined by a failure—the compiled premises do not close the goal—so a formalizer that rendered them into something inert would produce it for free. Both checks below live inside the obligation and use no reference, so they apply on every corpus.

Nothing is dropped. On all 17,080 accepted certificates of Table 6, the formalizer emits at least one premise clause for every premise sentence the candidate cited; there is no corpus and no step on which a cited premise silently disappears.

Inert premises are rare and do not carry the effect. Call an obligation inert when its compiled premises share no predicate symbol with its compiled conclusion. Under the original prompt this is 0.2–3.2% of accepted certificates, and excluding them moves the carried share by at most 0.014 (Table 7).

Table 7: The carried share of Table 6 (ORIG) with inert obligations excluded. "inert" = the compiled premises share no predicate symbol with the compiled conclusion.

Corpus	Accept.	Inert	Carried	Carried (non-inert)
ProntoQA	9,011	0.002	0.148	0.148
ProofWriter	2,798	0.017	0.357	0.351
FOLIO	1,777	0.021	0.478	0.472
ProverQA	1,980	0.029	0.465	0.465
Multi-LogiEval	989	0.024	0.408	0.400
e-SNLI	525	0.032	0.579	0.593

Predicate overlap tests for inertness, and must not be strengthened into a test for bridging. Requiring *every* conclusion predicate to occur in the premises drops the carried share to 0.019 on ProntoQA, but that condition selects against the phenomenon rather than against the artifact: the canonical laundered step is exactly one whose conclusion predicate the cited premises do not mention—`prem: vumpus(max), assum: vumpus(max) => tumpus(max), concl: tumpus(max)`—and on ProntoQA, where the corpus decides, 95% of the certificates on such steps are genuinely carried (Table 1).

What no reference-free check can exclude is a translation that looks faithful and is not, and on ProverQA none of 696 compiled obligations is equivalent to its source (Table 17). We therefore read the natural-language rows of Table 6 as an upper bound on laundering. Two things keep the bound from being vacuous. It is a bound on the interpretation, not on the structural claim: whatever the compiled premises mean, the pipeline accepted, the premises alone did not close the goal, the

candidate's own axiom did, and the certificate reports none of it. And the direction of the repair is wrong for an artifact. Three demonstrations make the compiler more faithful, and where the corpus lets us condition on it they *raise* the carried share on exactly the steps the claim is about: 0.952 to 0.991 on ProntoQA's correct locally unsupported steps and 0.774 to 0.828 on its incorrect ones (Table 1). The aggregate in Table 6 nonetheless falls slightly on the two program corpora (0.148 to 0.145, 0.357 to 0.332), because the repair also lifts acceptance (0.821 to 0.915, 0.420 to 0.538) and most of what it admits are steps whose premises already close the goal, where the carried rate collapses (0.016 to 0.001 on ProntoQA, 0.156 to 0.045 on ProofWriter); the aggregate is a mixture and moves with the mix. On the natural-language corpora that split is unavailable and the aggregate rises (Multi-LogiEval 0.408 to 0.617, e-SNLI 0.579 to 0.666). A carried share manufactured by bad compilation would fall under a better compiler in every one of these readings.

B.3 Policy, temperature and formalizer

Temperature and policy. The audited rollouts are sampled at temperature 1.3 from an 8B policy (§3), which invites the objection that the errors are manufactured and that a better-behaved policy would not launder. We repeat the chain on the same questions with the same formalizer, changing only the policy (Table 8). At temperature 0.7 the pool has 210 first-order-invalid steps against 221 and informedness 0.100 [.043,.160] against 0.087, and both provenance quantities move *against* the pipeline (carried share 0.899 against 0.774; unlicensed axioms 0.54 of false certificates against 0.40). A $4\times$ larger policy (Qwen3-32B-AWQ) is worse still, and its chains differ in the way the account predicts: it writes longer chains (6.8 steps per response against 4.5 for the 8B policy at the same temperature) in which each step cites its own previous conclusion rather than the problem's premises (53% of its cited entries against 12% for the 8B policy) and carries the rule in the assumption, so 94% of its valid steps are locally unsupported and informedness falls to 0.016 [−.021,.057], indistinguishable from zero. A more capable policy does not launder less. Its chains are longer and more often cite their own earlier conclusions, so more of its reasoning passes through the channel the assumption opens; the pools differ in several respects at once and the association does not identify a cause. One control we cannot fully resolve: with a policy that is not RLVR-trained (Mistral-7B-Instruct, Jiang et al., 2023) the structural leak replicates—0.788 of certificates on incorrect unsupported steps are carried by the candidate's assumption—but its prose defeats the corpus reader (49% of its carrying assumptions read, against 96% of the 8B policy's), so the unlicensed share among its false certificates is bounded only to [0.13, 0.44] and we claim no rate. What the control does establish is that being carried by a self-written axiom is not a property of reasoning-trained policies alone.

Table 8: The same 300 ProntoQA questions and the same Qwen3-8B formalizer under three policies. "unsupported" = share of first-order-valid steps whose cited premises do not entail the conclusion; "carried" = share of certificates on incorrect unsupported steps that the candidate's assumption carries; "laundered" = unlicensed axioms as a share of false certificates; "aligned" = share of steps whose every cited premise is a verbatim context sentence or a verbatim earlier conclusion, which rules out the reading that the 32B numbers come from worse step extraction—that pool is the best aligned. Laundering is a point where the reader leaves a negligible residual and a range otherwise: it reads 96% and 98% of the two 8B pools' carrying assumptions, 76% of the 32B pool's and 49% of the non-RLVR pool's ($*$), and the last two are therefore bounded. Released prompt (ORIG) throughout; the +DEMO row for the 32B pool reads -0.031 [-0.060, -0.000].

Policy	#Steps	Steps/resp.	Aligned	#Invalid	Inf	Carried	Unlicensed
Qwen3-8B, T=1.3 (main)	10,748	4.6	0.902	221	0.087 [.034,.159]	0.774	0.40
Qwen3-8B, T=0.7	10,551	4.5	0.891	210	0.100 [.043,.160]	0.899	**0.54**
Qwen3-32B, T=0.7	15,064	6.8	**0.982**	620	0.016 [−.021,.057]	0.971	0.50–0.67
Mistral-7B*, T=0.7	5,083	2.9	0.529	1,506	0.198 [.167,.228]	0.788	0.13–0.44

The closure-versus-first-order gap of §3 replicates on the fresh 8B pool (174 of 384 closure-invalid steps are first-order valid, against 179 of 400) and nearly vanishes for the 32B policy (16 of 636), which rarely reasons by contrapositive: the labelling correction matters most for weaker policies.

Formalizer. The pools above change the policy and keep the formalizer. Here we keep the main pools of both corpora and replace the formalizer by Qwen3-32B-AWQ, a $4\times$ larger model of the same family, under both prompts (Table 9). The larger compiler accepts more locally supported valid steps, 97% on ProntoQA against 89% and 65% on ProofWriter against 53%, and its certificates rest on the assumption more, not less: on unsupported steps the carried share rises to 0.99 and 0.82 on ProntoQA (from 0.95 and 0.77) and to 0.94 and 0.85 on ProofWriter (from 0.81 and 0.61), and the unlicensed share of false certificates does not fall. The released rule's informedness moves as the demonstration repair moved it, unchanged on ProntoQA (0.12 [.02,.19] against 0.09) and up on ProofWriter (0.26 against 0.15), and under either compiler the licensed and premises-only rules keep the same lead over it (0.48 and 0.42 on ProntoQA, 0.40 and 0.30 on ProofWriter). One change was made for this model and nothing else: it writes the field label as `**Logical form:**` with the formula on the following line, which the released parser's regular expression does not anticipate, so we strip the emphasis markers from that label before parsing.

Table 9: The main pools under two formalizers and two prompts. Accept.: acceptance on valid steps whose cited premises entail the conclusion; carried: share of accepted certificates on unsupported steps that the assumption carries, on valid and on invalid steps; informedness for the three rules with question-clustered 95% intervals; unlicensed: unlicensed axioms as a share of false certificates (a point where the reader parses nearly all carrying assumptions, a range otherwise).

				Carried		Informedness			
Corpus	Formalizer	Prompt	Accept.	valid	invalid	LogicReward	Licensed	Prem.-only	Unlic.
ProntoQA	Qwen3-8B (paper)	ORIG	0.891	0.952	0.774	0.087 [.034,.159]	0.410 [.332,.512]	0.339 [.253,.456]	0.40
ProntoQA	Qwen3-8B (paper)	+DEMO	0.988	0.991	0.828	0.091 [.043,.155]	0.486 [.417,.571]	0.420 [.338,.531]	0.44
ProntoQA	Qwen3-32B-AWQ	ORIG	0.973	0.992	0.817	0.116 [.024,.188]	0.483 [.408,.559]	0.419 [.335,.514]	0.42
ProntoQA	Qwen3-32B-AWQ	+DEMO	0.956	0.995	0.845	0.143 [.049,.215]	0.501 [.430,.581]	0.441 [.358,.539]	0.44
ProofWriter	Qwen3-8B (paper)	ORIG	0.531	0.812	0.612	0.150 [.098,.208]	0.220 [.164,.281]	0.145 [.101,.199]	0.25–0.39
ProofWriter	Qwen3-8B (paper)	+DEMO	0.725	0.922	0.760	0.250 [.185,.319]	0.374 [.308,.447]	0.287 [.226,.351]	0.28–0.49
ProofWriter	Qwen3-32B-AWQ	ORIG	0.645	0.935	0.853	0.262 [.200,.324]	0.395 [.334,.459]	0.301 [.248,.358]	0.35–0.60
ProofWriter	Qwen3-32B-AWQ	+DEMO	0.808	0.937	0.925	0.329 [.271,.386]	0.470 [.416,.529]	0.371 [.322,.423]	0.33–0.59

C ADDITIONAL RESULTS

C.1 REWARD AND FILTER TABLES

Tables 10 and 11 give the values plotted in Figures 2 and 3.

Table 10: Step-level informedness (left) and pairwise ranking accuracy of whole responses against the gold process score (right; 671 ProntoQA and 1,557 ProofWriter pairs from questions whose responses differ in that score), under first-order labels. Bootstrap 95% CIs resampling questions; best per column in bold. †: threshold τ chosen on the gold labels, which is not available at deployment. Licensed uses the corpus program as its reference and is a diagnostic, not a deployable rule (§4.3).

	Informedness		Pairwise ranking acc.	
Method	ProntoQA	ProofWriter	ProntoQA	ProofWriter
LogicReward (ORIG)	0.087 [.034,.159]	0.150 [.098,.208]	0.577 [.509,.646]	0.521 [.477,.565]
LogicReward (+DEMO)	0.091 [.043,.155]	0.250 [.185,.319]	0.573 [.524,.627]	0.548 [.509,.587]
Premises-only (ORIG)	0.339 [.253,.456]	0.145 [.101,.199]	0.697 [.640,.754]	0.539 [.500,.580]
Premises-only (+DEMO)	0.420 [.338,.531]	0.287 [.226,.351]	**0.770** [.718,.822]	0.600 [.554,.645]
Licensed (ORIG)	0.410 [.332,.512]	0.220 [.164,.281]	0.685 [.624,.751]	0.560 [.516,.605]
Licensed (+DEMO)	**0.486** [.417,.571]	**0.374** [.308,.447]	0.769 [.717,.824]	**0.630** [.581,.675]
Qwen2.5-Math-PRM-7B (τ=0.5)	0.229 [.124,.373]	0.092 [.050,.139]	0.595 [.529,.661]	0.582 [.532,.634]
same, oracle $\tau^\dagger$	0.360	0.256	—	—
Length baseline	—	—	0.592 [.527,.659]	0.538 [.483,.593]

Table 11: The certificate as a filter over whole responses, at its own threshold: accept a response only if *every* step is certified, so there is no free parameter. Yield, answer accuracy and share of conclusion-valid steps. $*$: requires the corpus program, so it is not comparable with the two baselines, which receive no reference. The learned PRM and the length baseline have no such threshold and are taken at the yield of Premises-only (+DEMO). Best answer accuracy among filters in bold. Paired over ProofWriter responses with questions resampled, Premises-only (+DEMO) beats the PRM by $+0.133$ [.054,.237] at that yield; its advantage over the length baseline, $+0.098$ [$-$.006,.231], does not exclude zero. Licensed (+DEMO) beats both, by $+0.171$ [.092,.249] and $+0.111$ [.026,.211].

Method	ProofWriter (n = 1,064) Yield	Answer acc.	Step validity	ProntoQA (n = 2,382) Yield	Answer acc.	Step validity
No filter	1.000	0.722	0.819	1.000	0.980	0.983
LogicReward (ORIG)	0.148	0.860	0.887	0.528	0.983	0.986
LogicReward (+DEMO)	0.246	0.851	0.908	0.792	0.984	0.985
Licensed (ORIG)*	0.102	**0.945**	0.966	0.496	0.989	0.996
Licensed (+DEMO)*	0.179	0.942	0.987	0.736	0.991	0.996
Premises-only (ORIG)	0.078	0.928	0.976	0.431	0.988	0.996
Premises-only (+DEMO)	0.134	0.930	**0.988**	0.645	0.990	0.996
Qwen2.5-Math-PRM-7B, matched yield	0.134	0.797	0.942	0.645	0.977	0.989
Length baseline, matched yield	0.134	0.832	0.857	0.645	**0.993**	0.984

C.2 ASSUMPTION CLASSES AND THE SECOND DEFECT

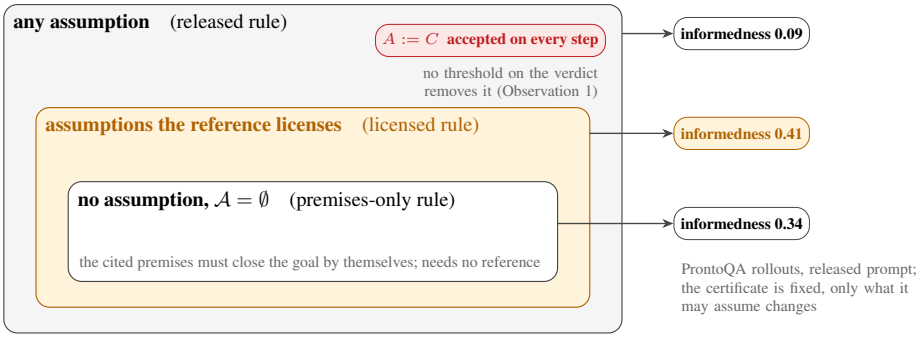

Figure 4: What the certificate may assume. The released rule admits any assumption, including the conclusion itself, which is accepted on every step (Observation 1); the licensed rule admits only assumptions the reference entails; the premises-only rule admits none. Informedness on the ProntoQA rollouts under the released prompt, with the prover, compiler and threshold held fixed.

Placebo demonstrations and instruction. We cross the rule stated in prose with three worked examples whose subjects are either named or—as a count- and form-matched placebo—kinds, on the held-out audit (n = 3,031, Table 18): three kind-subject demonstrations change nothing (0.690 / 0.590, within the formalizer's own sampling variation), as predicted. The instruction alone moves ProntoQA from 0.661 to 0.800 and leaves ProofWriter unchanged (0.585→0.563); three named-subject demonstrations take both corpora to 0.996–1.000 and 0.876–0.884 with invalid acceptance 0.000–0.004. Placebo demonstrations plus the instruction reach 0.916 / 0.684, so a stated rule takes effect only when a worked example carries it.

Compilers whose demonstrations show names. Run on the same sentences (Table 12), only the compiler whose demonstrations contain no named subject renders entities as predicates (0.106–0.291 against at most 0.040)—observational on its own; the injection experiment of §6 is what makes it an attribution. The defect, unlike laundering, is a property of any learned compiler whose demonstrations omit named subjects, whether or not it permits assumptions.

Table 12: Cross-implementation control: rate at which a named individual is rendered as a predicate, one model, same sentences, conclusion formula only. Only the pipeline whose demonstrations contain no named subject shows the defect. The LINC extractor yields 0.250 spurious "absent" and never a false "predicate", so the contrast is conservative.

Corpus	LogicReward (0/8 named demos)	Logic-LM (13/13 kept)	LINC (gold FOL demos)
ProofWriter	**0.284**	0.000	0.000
ProntoQA	**0.106**	0.000	0.000
ProverQA	**0.291**	0.000	0.000
FOLIO	**0.137**	0.040	0.032

Fidelity of the repaired obligation. Informedness could rise because the obligation becomes right or because it becomes unprovable in the wrong direction; bidirectional equivalence against the corpora's ground truth (Table 17) separates them. Three demonstrations raise obligation equivalence from 0.551 to 0.840 and 0.435 to 0.828, and verdict preservation to 1.000 and 0.970; a neutral-vocabulary variant reaches ProverQA informedness 0.125 against 0.136, so the effect is structural, not lexical. The repair replicates across families (GLM-4-9B 0.787→1.000 and 0.613→0.840; gpt-oss-20b the direction only, 0.067→0.240 and 0.173→0.267). The defect is not in the prover or the scale of the formalizer: Vampire and E return the same verdict on all 700 ProntoQA obligations, and a $4\times$ larger formalizer (Qwen3-32B-AWQ) under the released prompt accepts wrong-entity siblings on ProntoQA at 0.743 against the 8B model's 0.749.

Table 13: What the carrying assumption is, for assumption-carried certificates of the original formalizer, read against the question's own program. "Licensed" = the program first-order entails it; "Unlicensed" = it parses and the program does not entail it; "Unparsed" = the reader cannot map the sentence onto a formula. The last column bounds the unlicensed share by assigning every unparsed assumption first to the licensed and then to the unlicensed side. Wilson 95% CIs.

Corpus	Gold	#	Licensed	Unlicensed	Unparsed	Unlicensed (bounds)
ProntoQA	valid	1,124	**0.910** [.892,.925]	0.056 [.044,.071]	0.034 [.025,.046]	0.06–0.09
ProntoQA	invalid	82	0.098 [.050,.181]	**0.817** [.720,.886]	0.085 [.042,.166]	**0.82–0.90**
ProofWriter	valid	459	**0.734** [.692,.773]	0.063 [.044,.089]	0.203 [.168,.242]	0.06–0.27
ProofWriter	invalid	142	0.162 [.110,.231]	**0.535** [.453,.615]	0.303 [.233,.383]	**0.54–0.84**

Table 14: Held-out constructed audit ($n = 3{,}037$), Qwen3-8B formalizer. Acceptance on valid (A^+) and invalid (A^-) steps and informedness $A^+ - A^-$, for the original prompt (ORIG), the original prompt with three named-subject demonstrations (+DEMO), and a deterministic constant contract applied to the released output with no model call (CONTRACT).

		ORIG		+DEMO		CONTRACT	
Corpus	# (valid/invalid)	A^+	A^-	A^+	A^-	A^+	A^-
ProntoQA	175/525	0.891	0.250	**1.000**	**0.000**	0.989	0.006
ProofWriter	225/675	0.724	0.181	**0.880**	**0.000**	0.849	0.019
FOLIO	225/225	0.133	0.031	0.196	0.022	0.151	0.040
e-SNLI	275/275	0.171	0.095	0.211	0.135	0.171	0.095
informedness:	ProntoQA	0.642		**1.000**		0.983	
	ProofWriter	0.544		**0.880**		0.830	
	FOLIO	0.102		0.173		0.111	
	e-SNLI	0.076		0.076		0.076	

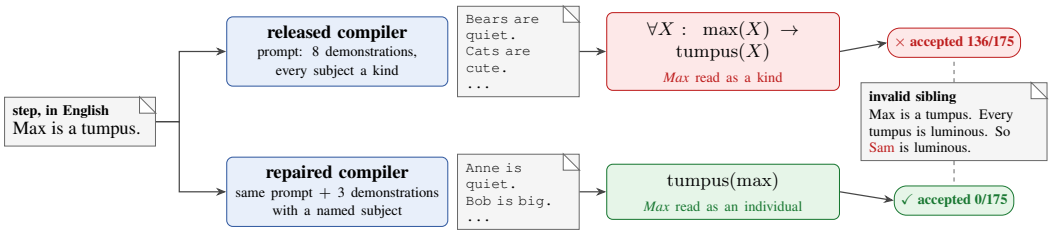

Figure 5: The second defect on one sentence. The released compiler, whose eight demonstrations all have a kind as subject, renders the name *Max* as the predicate $\max(X)$; a statement about that kind is weaker than one about the individual, and any rule that mentions it closes the obligation, so an invalid sibling that applies the right rule to the wrong individual is accepted. Adding three demonstrations with a named subject renders the name as a constant and the same siblings are refused (ProntoQA, held-out ablation run of Table 18: 136/175 against 0/175; ProofWriter 105/225 against 0/225).

Table 15: Invalid acceptance by construction of the invalid sibling (held-out). The failure is one mutation.

Corpus	Construction	#	ORIG	CONTRACT
ProntoQA	wrong entity	175	**0.749**	0.017
ProntoQA	affirming the consequent	175	0.000	0.000
ProntoQA	denying the antecedent	175	0.000	0.000
ProofWriter	wrong entity	225	**0.498**	0.013
ProofWriter	affirming the consequent	225	0.000	0.000
ProofWriter	denying the antecedent	225	0.044	0.044

Table 16: How proper names in the source are rendered in the original formalization, counted with no prover and no TPTP conversion (FOLIO 0.087 and e-SNLI 0.396 predicate by a capitalisation heuristic, indicative only).

Corpus	#Name occurrences	Constant	Predicate	Absent
ProntoQA	1,400	0.838	**0.127**	0.035
ProofWriter	1,954	0.614	**0.301**	0.085

Table 17: Fidelity against the corpora's symbolic ground truth: is the compiled conclusion / whole obligation bidirectionally equivalent (BEq, Poiroux et al., 2025) to the gold formula, and is the prover's verdict on the gold obligation preserved?

Corpus	Quantity	#	ORIG	+DEMO
ProntoQA	conclusion equivalent	700	0.587	**0.901**
ProntoQA	obligation equivalent	700	0.551	**0.840**
ProntoQA	verdict preserved	700	0.786	**1.000**
ProofWriter	conclusion equivalent	898	0.477	**0.958**
ProofWriter	obligation equivalent	898	0.435	**0.828**
ProofWriter	verdict preserved	898	0.797	**0.970**
ProverQA	obligation equivalent	696	0.000	0.009
ProverQA	verdict preserved	696	0.734	0.797

Table 18: Attribution on the held-out audit ($n = 3{,}031$), informedness and invalid acceptance. Rows: the original prompt, plus the instruction, plus three demonstrations with kind subjects (placebo), plus three with named subjects, and their combinations. The formalizer is sampled and its verdicts vary by about 5% across repetitions, so the ORIG row here reads 0.661 / 0.585 against 0.642 / 0.544 in Table 14, within both confidence intervals.

	ProntoQA		ProofWriter	
Variant	Inf	A^-	Inf	A^-
ORIG	0.661	0.259	0.585	0.179
+ instruction	0.800	0.189	0.563	0.184
+ 3 kind-subject demos (placebo)	0.690	0.288	0.590	0.179
+ 3 kind-subject demos + instruction	0.916	0.084	0.684	0.133
+ 3 named-subject demos	**0.996**	**0.004**	**0.876**	**0.000**
+ 3 named-subject demos + instruction (+DEMO)	**1.000**	**0.000**	**0.884**	**0.000**

C.3 NATURAL-LANGUAGE CORPORA

On FOLIO, e-SNLI, ProverQA and Multi-LogiEval the released pipeline refuses 83–87% of valid natural-language steps (informedness 0.076–0.173; Multi-LogiEval -0.024), scaling makes it worse (e-SNLI valid acceptance 0.171→0.015 from 8B to 32B), the named individual disappears from the formalization 60–84% of the time, and the three-demonstration repair moves FOLIO from 0.102 to 0.173 and nothing else.

C.4 THE FILTER ON CORPORA WITHOUT A PROGRAM

The premises-only rule needs no program, so it can be applied on the four corpora of Table 6 that ship none, all of which have headroom in answer accuracy (base 0.629–0.789). The licensed rule cannot: without a program nothing decides whether an assumption is licensed, and nothing decides step validity either, so answer accuracy is the only outcome left.

Selection quality replicates, but in a different regime (Table 20). Yields fall an order of magnitude below ProofWriter's 13.4%—requiring every step to be certified keeps 1.3–3.5% of responses under premises-only—and what survives is almost always a one-step response (1.0–1.5 steps against a base of 2.0–4.7). The filter is therefore partly a length filter, and the comparison that controls for it is the length baseline at matched yield, which selects the same number of the shortest responses and so the same length regime. Against that baseline the gain separates from zero on two of the four corpora, e-SNLI ($+0.121$ [.036,.239]) and FOLIO under +DEMO ($+0.261$ [.083,.459]); on ProverQA ($+0.087$ [$-$.080,.341]) and Multi-LogiEval ($+0.049$ [$-$.017,.162]) it does not, and we claim nothing there. The original certificate does not survive the same control, losing to the length baseline on FOLIO and Multi-LogiEval.

None of this says the certificate means what it claims on these corpora. Appendix C.3 shows it does not, and on ProverQA almost no obligation is equivalent to its source (Table 17). It says that the constrained rule selects better than the unconstrained one even where the compiler is unfaithful. What it selects are short, local steps, and on these corpora such responses are more often correct; the comparison identifies an operating point, not a mechanism.

Best-of-N selection. The standard inference-time use of a process reward is best-of-N: score every sample and keep the top one. Table 19 applies each rule that way over the eight samples per question on all six corpora, resolving the ties that coarse certificate shares produce by averaging over the tied-best set (the expected accuracy of a random tie-break). No reference-free rule beats random selection by more than 0.022 anywhere, and against the shortest-response baseline no interval excludes zero in the rule's favour except one marginal cell (Multi-LogiEval, released, $+0.029$ [.001,.059]). The learned PRM does no better. This is the filter result seen from the other side. The certificate's information is concentrated in the tail of responses where *every* step is certified: a threshold on that tail is a usable curation filter, and ordering responses by the share of certified steps tracks process quality (§5.3) without tracking which response ends in the right answer. The two are different targets, and only the first is what the certificate scores.

Table 19: Best-of-8 answer accuracy by selection rule on six corpora; "random" is the corpus base, and the PRM exists on the two program corpora only. The premises-only rule (+DEMO) minus shortest, paired bootstrap over questions, top to bottom: $-.003$ [$-.015$,.009], $+.033$ [$-.031$,.099], $-.002$ [$-.044$,.040], $+.029$ [$-.020$,.075], $+.003$ [$-.024$,.031], $-.004$ [$-.023$,.013].

Corpus (#)	Random	LogicReward ORIG	Prem.-only ORIG	Prem.-only +DEMO	Shortest	PRM
ProntoQA (300)	0.980	0.981	0.984	0.987	**0.990**	0.977
ProofWriter (150)	0.726	0.728	0.696	0.733	0.700	**0.747**
ProverQA (293)	0.681	**0.686**	0.680	0.667	0.669	—
FOLIO (204)	0.650	0.652	0.664	**0.666**	0.637	—
Multi-LogiEval (299)	0.789	**0.810**	0.791	0.784	0.781	—
e-SNLI (300)	0.629	0.630	**0.641**	0.635	0.640	—

Table 20: The filter at its own threshold—accept a response only if *every* step is certified—on the four corpora that ship no program. "steps" is the mean number of steps in a kept response, against the corpus base in the header row; "length-m." is the length baseline taking the same number of the shortest responses; "gap" is rule minus that baseline, with a 95% bootstrap interval resampling questions (2,000 resamples, both selections recomputed inside each resample). Rollouts are Qwen3-8B at $T = 0.7$, eight samples per question, the same formalizer throughout.

Method	Yield	#Steps	Answer acc.	Length baseline	Gap [95% CI]
ProverQA — $n = 2{,}313$ responses, base answer 0.680, 4.69 steps					
LogicReward, ORIG	0.042	1.14	0.865	0.740	$+0.125$ [.026,.223]
prem.-only, ORIG	0.020	1.04	0.891	0.804	$+0.087$ [$-.080$,.341]
prem.-only, +DEMO	0.022	1.21	0.865	0.788	$+0.077$ [$-.088$,.321]
FOLIO — $n = 1{,}632$ responses, base answer 0.650, 3.95 steps					
LogicReward, ORIG	0.080	1.69	0.733	0.779	-0.046 [$-.102$,.071]
prem.-only, ORIG	0.028	1.13	0.957	0.739	$+0.217$ [.000,.417]
prem.-only, +DEMO	0.028	1.26	1.000	0.739	$+\mathbf{0.261}$ [.083,.459]
Multi-LogiEval — $n = 2{,}388$ responses, base answer 0.789, 3.29 steps					
LogicReward, ORIG	0.026	1.25	0.889	0.937	-0.048 [$-.127$,.060]
prem.-only, ORIG	0.013	1.47	0.933	0.900	$+0.033$ [$-.160$,.174]
prem.-only, +DEMO	0.026	1.07	0.984	0.934	$+0.049$ [$-.017$,.162]
e-SNLI — $n = 2{,}398$ responses, base answer 0.629, 2.03 steps					
LogicReward, ORIG	0.060	1.11	0.916	0.895	$+0.021$ [$-.026$,.112]
prem.-only, ORIG	0.024	1.02	1.000	0.879	$+\mathbf{0.121}$ [.036,.239]
prem.-only, +DEMO	0.035	1.01	1.000	0.880	$+\mathbf{0.120}$ [.046,.219]

C.5 A JUDGE MODEL IN PLACE OF THE CORPUS READER

Obligations (1)–(3) run anywhere; readings (4)–(5) need a corpus that ships its program. To see what is lost elsewhere we replace the reader with the substitute a practitioner would reach for—a judge model shown the context and asked whether it licenses the assumption, in three configurations—calibrated against the reader on the 4,140 carried certificates of the two program corpora (ORIG and +DEMO pooled). The ordering is usable but configuration-dependent (AUROC 0.77–0.87 on ProofWriter, 0.57–0.89 on ProntoQA); the rate is not usable at all, the three judges reporting 0.028–0.185 against the reader's 0.124 on ProntoQA and 0.088–0.157 against 0.241 on ProofWriter. A reference-free audit can rank assumptions for review, but its headline rate moves sixfold with the judge.

D Other Pipelines and Training

D.1 VeriCoT-style verification

VeriCoT-style verification (Feng et al., 2026) types every premise and refuses a step whose premises are not grounded in the context: the licensing condition enforced at the premise level rather than at the goal. Table 21 reports it with and without its natural-language judge. On the same steps and labels it behaves as the condition predicts, in both directions. On ProntoQA it reaches informedness 0.257 [.195,.316] against 0.087, with the channel the assumption opens nearly closed: 3.5% of its accepted certificates use a model-written premise. The price is refusals, and on ProofWriter they dominate: 85% of steps are called ungrounded and informedness falls to 0.053 [.033,.072], below the original certificate's 0.150. Constraining what may be assumed buys resistance to laundering and costs coverage.

Table 21: Our reimplementation of VeriCoT-style verification on the rollout steps, scored against the same first-order oracle, with and without its natural-language judge; bootstrap 95% CIs on informedness. "Model-written": share of accepted certificates that use at least one premise the model supplied rather than a context sentence. The released LogicReward certificate on the same steps is 0.087 (ProntoQA) and 0.150 (ProofWriter).

Corpus	Method	A^+	A^-	Inf	Ungrounded	Model-written
ProntoQA	with judge	0.551	0.294	0.257 [.195,.316]	0.428	0.035
ProntoQA	without judge	0.547	0.330	0.217 [.156,.279]	0.429	0.037
ProofWriter	with judge	0.123	0.070	0.053 [.033,.072]	0.848	0.000
ProofWriter	without judge	0.267	0.137	0.129 [.103,.155]	0.721	0.077

D.2 Matched DPO

Whether selecting training data with the certificate improves a trained policy is the question the filter of §5.3 raises, and we do not settle it. On a pool of 450 ProofWriter validation questions disjoint from the audit, with eight rollouts per question, certificates under both prompts and first-order oracle labels, we build preference pairs within a question under six selection rules: the original certificate, the repaired certificate, premises-only under each prompt, final-answer correctness, and the gold process score. Pairs are restricted to the 188 questions on which every rule decides, so every arm trains on the same number of pairs from the same questions. Each arm is trained three times (LoRA rank 16, $\beta = 0.1$, learning rate 5×10^{-6}, one epoch) and decoded greedily on 700 held-out pool questions and 500 held-out ProofWriter questions for answer accuracy, and on 200 ProntoQA test questions and the 150 audit ProofWriter questions for step validity, scored by the same oracles as the rest of the paper.

No arm separates from the base policy on any measure (Table 22). Across the 36 paired comparisons the widest gap is $+0.03$ and every 95% interval contains zero, the gold-process arm included; on ProntoQA claim validity no gap exceeds $+0.004$. The training explains why: the DPO loss ends within 0.01 of $\ln 2$ for every arm and the final reward margin is within ± 0.05, so 188 pairs at this budget do not move the policy under any rule, and the comparison has no power to rank the rules. What the pool does show is at the pair level: on the pairs the oracle also decides, the original certificate orders the two responses as the oracle does on 64 and inverts it on 43, premises-only on 64 and 51, and final-answer correctness on 91 and 41. A powered comparison needs a budget that moves the policy under the oracle before the rules can be told apart.

Table 22: Matched LoRA-DPO from one rollout pool under six selection rules, 188 pairs per arm, three seeds pooled. Each cell is the arm minus the base policy, paired by question, with a 95% bootstrap interval resampling questions; the base row gives absolute values. Answer accuracy on 700 held-out pool questions and on 500 held-out ProofWriter questions; on the 150 audit ProofWriter questions, mean step validity per response and the share of responses whose every mapped step is valid. No interval excludes zero.

Selection rule	Answer, 700	Answer, PW-500	PW step valid.	PW all valid
Base policy (absolute)	0.676	0.726	0.849	0.684
LogicReward (ORIG)	+.005 [−.013,.025]	+.007 [−.022,.034]	+.010 [−.013,.038]	+.029 [−.018,.080]
LogicReward (+DEMO)	+.001 [−.020,.022]	+.019 [−.011,.047]	+.011 [−.014,.037]	+.028 [−.021,.080]
Premises-only (ORIG)	+.003 [−.016,.025]	−.000 [−.029,.028]	+.003 [−.018,.025]	+.012 [−.035,.062]
Premises-only (+DEMO)	−.010 [−.030,.011]	+.013 [−.017,.040]	+.015 [−.008,.041]	+.024 [−.027,.079]
Final answer	−.009 [−.028,.012]	+.007 [−.021,.033]	+.013 [−.008,.036]	+.027 [−.018,.073]
Gold process score	−.001 [−.021,.020]	+.019 [−.009,.046]	+.012 [−.015,.041]	+.009 [−.047,.069]