IS IT A RANKING FAILURE? A SAME-BUDGET ACTION-ORDERING AUDIT FOR VERIFIER-GUIDED LOGICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

When a verifier-guided reasoning system fails within a fixed budget, can changing only the order of its actions certify the correct answer? We introduce a same-budget audit that answers this question while holding the rules, available actions, verifier, and stopping rule fixed. For finite grounded Horn-style forward chaining, shortest certifying distances define the fraction of instances with some successful ordering within budget. Offline search and saturation-cost arguments determine this ceiling, with explicit bounds when resource-capped search leaves cases unresolved. On 21,677 instances from a restricted ProofWriter test subset and generated PrOntoQA partitions, all 4,988 selected-policy failures at an eight-proposal budget require saturation: the stopping rule waits until every derivable fact has been obtained. Their exact costs are 9–19 proposals, so reordering cannot repair any of these failures: the selected policies attain the ceiling. Lower-budget audits also identify repairable failures, including one ProofWriter instance missed by all thirteen tested policies. Broader data scope, changed rule order, and alternative stopping rules reveal the limits of the zero-gap finding. The audit distinguishes failures caused by action ordering from those constrained by the fixed certification protocol, providing a basis for deciding which component to improve.

1 INTRODUCTION

Verifier-guided reasoning separates proposing a step, checking its validity, and deciding when an answer has been certified. This pattern appears in natural-language proof search and tool-guided deduction (Yang et al., 2022; Poesia et al., 2024). Learned guidance can improve bounded proof search (Lample et al., 2022), making a stronger ranker a plausible response to failures. Before changing the ranker, we ask whether a different order of the same available actions could have succeeded within the same number of proposals.

Consider two instances from our PrOntoQA evaluation. In the first, we know `yumpus(Alex)` and can prove `happy(Alex)` in two steps:

$$\texttt{yumpus(Alex)} \rightarrow \texttt{impus(Alex)} \rightarrow \texttt{happy(Alex)}.$$

After deriving `impus(Alex)`, one ordering chooses another legal consequence, `hot(Alex)`, and runs out of its two-step budget. Choosing `happy(Alex)` as the second action succeeds. This failure can be repaired by changing the second action. In the second instance, the query is `large(Sally)` and the answer is NO. Our stopping rule requires deriving all consequences before certifying this answer. There are twelve new facts to derive, each costing one proposal. Every ordering therefore fails with a budget of eight. The difference is whether a successful ordering fits the budget at all. Appendix E gives both instances.

A *ranker* scores and reorders available candidate actions, keeping the candidate set fixed. We call a failure *ranker-repairable* if some ordering certifies the correct answer within the same proposal budget while the rules, candidates, transitions, verifier, and stopping rule remain fixed. A finite policy comparison can demonstrate a repair by finding a successful alternative. When all tested policies fail, however, other orderings remain unexamined. Establishing whether a successful ordering exists is the question our audit addresses.

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.

We study finite grounded Horn-style forward chaining, where every currently fireable rule with a new head is available and each accepted proposal adds one new literal. A shortest path to an answer-certifying state gives the minimum number of proposals needed for an instance. The fraction of instances whose shortest distance fits the budget is the *same-budget action-ordering ceiling*. For answers certified only after all consequences have been derived, classical fixed-point semantics (van Emden & Kowalski, 1976) give a simpler calculation: every ordering must add the same new literals, so their count is the exact certification cost. The audit uses successful trajectories, these costs, and exact search to identify which failures admit a within-budget repair. Unfinished searches retain explicit uncertainty bounds.

Our primary evaluation combines a restricted ProofWriter test subset with PrOntoQA evaluation partitions. At eight proposals, all selected-policy failures in these 21,677 instances are UNKNOWN or NO answers with costs of 9–19 proposals under the fixed stopping rules. Successful trajectories and these failure costs establish a zero ordering gap on this cohort. At lower budgets, query-directed ordering improves PrOntoQA certified accuracy by 49.97–51.81 percentage points through earlier YES proofs. A four-proposal ProofWriter audit finds 54 repairable failures, including one missed by all thirteen tested configurations. Restoring excluded negative-premise questions and perturbing rule-index tie order also reveal small nonzero gaps at eight proposals. These comparisons locate when ordering changes the outcome and when certification costs prevent it from doing so.

Contributions. We formulate an ordering-only repairability test with explicit fixed components and resource-cap accounting. We establish the primary zero gap by combining observed failure classes with exact certification costs, and use a witness beyond the tested policy portfolio to demonstrate the added information from shortest-path analysis. Controlled changes to budget, data scope, rule order, and stopping rules identify the conditions under which the diagnosis changes. The resulting audit supports choosing an intervention based on the source of a failure.

2 RELATED WORK

Logical evaluation and goal-directed search. The fixed-point semantics of Horn programs are classical (van Emden & Kowalski, 1976). Our saturation identity specializes this foundation to unit-cost steps that add one novel literal and a stopping rule that requires the full closure. Magic-set transformations reduce irrelevant derivations by rewriting a program around a query (Bancilhon et al., 1986). Such transformations change the rule system, whereas our audit fixes it and varies only action ordering. Whether saturation is required depends on the stopping rule; Section 4.5 measures that dependence.

Verifier-guided and neuro-symbolic reasoning. Natural-language deduction can be decomposed into steps coordinated by search (Bostrom et al., 2022). NLProofS scores candidate steps with an independent verifier (Yang et al., 2022); LEAP uses lookahead planning and learned verification (Zhao et al., 2023). Logic-LM translates problems for symbolic solvers (Pan et al., 2023), while LogicGuide constrains generation to formally licensed deductions (Poesia et al., 2024). Other work verifies complete reasoning texts (Pei et al., 2025) or individual deductive steps (Sadeddine & Suchanek, 2025). RuleTaker, ProofWriter, and PrOntoQA offer controlled rule-deduction settings (Clark et al., 2020; Tafjord et al., 2021; Saparov & He, 2023); FOLIO and LogicBench cover broader logical problems (Han et al., 2024; Parmar et al., 2024). Our experiments isolate action ordering through symbolic agendas and exact verification; candidate generation and learned verification are held outside the intervention.

PrOntoQA's proof analysis already distinguishes validity from utility relative to a gold proof (Saparov & He, 2023, Section 4): a locally valid step can leave that proof. For an entire failed rollout, we ask whether *any* accepted-action ordering certifies the answer within the same budget. This budget-conditioned existence test ranges over alternative paths, rather than agreement with a reference proof, and can also certify non-repairability under explicitly fixed components.

Learned proof guidance and inference-time computation. CoqGym, HOList, HyperTree Proof Search, LeanDojo, and Magnushammer illustrate learning to guide proof construction or premise selection (Yang & Deng, 2019; Bansal et al., 2019; Lample et al., 2022; Yang et al., 2023; Mikuła et al., 2024). Self-consistency, process verifiers, dynamic-tree sampling, and recurrent depth allocate

inference computation in other ways (Wang et al., 2023b; Setlur et al., 2025; Wang et al., 2023a; Geiping et al., 2025). These works motivate examining how computation is allocated. We isolate the effect of reordering within a fixed symbolic action space.

3 A SAME-BUDGET ACTION-ORDERING AUDIT

3.1 FIXED SYSTEM AND CERTIFICATION MODES

An instance is $x = (F_0, \mathcal{R}_x, q, y_x)$: initial ground literals, a finite ground definite Horn-style rule set, a query, and its gold answer. A rule $r = (\mathrm{body}(r), \mathrm{head}(r))$ has finitely many literal premises and one head. Explicit negation is represented as a separate literal; negation as failure and nonmonotonic revision are excluded. At fact set F, the *agenda* contains every rule whose premises are present and whose head would be new:

$$A(F) = \{r \in \mathcal{R}_x : \mathrm{body}(r) \subseteq F,\ \mathrm{head}(r) \notin F\}. \tag{1}$$

The evaluated policies choose among these already-legal actions. The local verifier checks the chosen rule and applies its state transition; it does not search for a proof. An accepted proposal adds exactly one new literal, $F' = F \cup \{\mathrm{head}(r)\}$; a rejected proposal preserves F. Both consume one unit of proposal budget B. Thus all evaluated policy proposals are accepted, although the general bound also allows proposals outside $A(F)$. Rule applicability and stopping are determined by F and the fixed rule set, so the analysis can use F alone as the state. Write $s_x(F)$ for the reconstructed runtime state. A policy may retain its own history, such as queue order; this does not change the transitions or stopping labels determined by F. Appendix A gives the runtime representation.

The state-based stopping rule $C_x(s)$ is checked initially, before each proposal, and after the last allowed proposal. It returns a label or $\perp$ (undecided). For ProofWriter, our open-world rule returns TRUE or FALSE upon deriving q or its explicit opposite $\bar{q}$, and UNKNOWN only after agenda exhaustion with neither derived. For PrOntoQA, our fixed closed-world rule returns YES on deriving q, and NO only after agenda exhaustion without q. We retain this saturation-only negative stopping rule for the primary audit. It is a protocol choice, not a requirement of every valid PrOntoQA solver; Section 4.5 separately allows earlier stopping on $\bar{q}$.

Witness certification stops when the query, or an opposite permitted by the stopping rule, is derived; *saturation certification* waits until no accepted action remains. The Alex and Sally examples in Section 1 illustrate these two modes. The first reaches a query witness; the second must exhaust the agenda before its negative answer is certified.

Assumptions. The guarantees require (i) unit-cost proposals with strict one-literal extension on acceptance and no state change on rejection; (ii) exact agenda maintenance; (iii) a sound, state-based stopping rule, meaning $C_x(s) \neq \perp \Rightarrow C_x(s) = y_x$ for every reachable state; and (iv) policies sharing the rule set, agenda construction, transitions, verifier, stopping rule, and budget, differing only in action selection. There is no probabilistic assumption on the policy. Stopping-rule soundness is a semantic precondition, distinct from checking a returned trajectory.

3.2 DISTANCE, CEILING, AND REPAIRABLE FAILURES

Let $s_t^\pi(x)$ be the state after t proposals. Certified accuracy is

$$\mathrm{CA}_B(\pi; X) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}[\exists t \in \{0, \ldots, B\} : C_x(s_t^\pi(x)) = y_x]. \tag{2}$$

The denominator includes every evaluation instance. In the *complete accepted-state graph* $\mathcal{G}_x$, nodes are reachable fact sets and edges are accepted actions. Its unit-cost edges are $F \to F \cup \{\mathrm{head}(r)\}$ for every $r \in A(F)$.

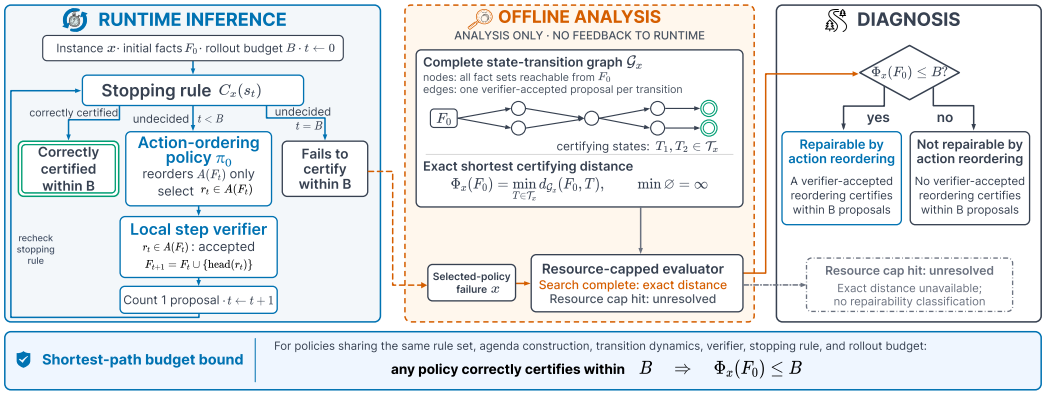


Figure 1: Runtime inference and search-based diagnosis of a selected-policy failure. The depicted state-transition graph is the complete accepted-state graph: nodes are reachable fact sets and edges are accepted actions. The failure branch illustrates diagnosis, not the scope of evaluation: offline searches also include selected-policy successes and never guide inference. A resource cap leaves repairability unresolved by that search alone. Successful trajectories or saturation costs can still determine whether certification fits the budget; these two evidence sources suffice for the primary eight-proposal zero-gap result.

Let $s_x(F)$ reconstruct the runtime state and let $\mathcal{T}_x = \{F : C_x(s_x(F)) = y_x\}$ be its certifying nodes. Define the shortest certifying distance and the distance-decreasing actions by

$$\Phi_x(F) = \min_{T \in \mathcal{T}_x} d_{\mathcal{G}_x}(F, T), \tag{3}$$

$$A_x^+(F) = \{r \in A(F) : \Phi_x(F \cup \{\text{head}(r)\}) = \Phi_x(F) - 1\}. \tag{4}$$

If no certifying terminal is reachable, set $\Phi_x(F) = \infty$; for zero or infinite distance, set $A_x^+(F) = \varnothing$. Validity of a step is local; membership in $A_x^+(F)$ concerns progress along a shortest certifying path.

Three distinct judgments. The verifier checks step legality, the stopping rule checks whether the state certifies an answer, and the offline evaluator computes the fewest remaining accepted proposals needed (Figure 1). A legal action outside $A_x^+(F)$ need not help within budget; saturation-only instances instead admit many legal orders with the same cost.

The *same-budget action-ordering ceiling* and the gap of a policy are

$$\text{CA}_B^{\text{ord}\star}(X) = \frac{1}{|X|} \sum_{x \in X} \mathbf{1}[\Phi_x(F_0) \le B], \tag{5}$$

$$\Delta_B^{\text{ord}}(\pi; X) = \text{CA}_B^{\text{ord}\star}(X) - \text{CA}_B(\pi; X). \tag{6}$$

Thus Φ_x is a distance for one instance, whereas the ceiling is a fraction of the evaluation set. Each instance may use a different successful ordering; the ceiling does not identify a single deployable policy. For the selected policy π_0, write $Q_B = \{x : \Phi_x(F_0) \le B\}$ and $P_B = \{x : \pi_0 \text{ correctly certifies } x \text{ within } B\}$. The repairable failures are $R_B = Q_B \setminus P_B$, and $\Delta_B^{\text{ord}}(\pi_0; X) = |R_B|/|X|$.

Proposition 1 (Policy-independent budget bound)**.** *Under the four assumptions in Section 3.1, every action-selection policy π, including a learned ranker, satisfies*

$$\pi \text{ correctly certifies } x \text{ within } B \quad \Longrightarrow \quad \Phi_x(F_0) \le B. \tag{7}$$

Proof. Deleting rejected proposals from a successful rollout preserves its state changes and final label. The remaining accepted actions form a path to a certifying node of length at most B, so the shortest such path has length at most B. □

Conversely, a certifying path of length at most B can be followed without rejections. Under sound state-based stopping, any earlier termination is also correct. Thus a successful ordering exists exactly when $\Phi_x(F_0) \le B$. The bound gives $P_B \subseteq Q_B$ and a nonnegative gap.

3.3 SATURATION COSTS AND RESOURCE-CAPPED SEARCH

Let $\mathrm{Cl}_x(F_0)$ denote the least closure obtained by repeatedly applying the rules until no new literal can be derived. Its fixed-point semantics are standard (van Emden & Kowalski, 1976). When the stopping rule certifies the gold label at saturation and never earlier, our unit-cost accounting yields

$$\Phi_x(F_0) = |\,\mathrm{Cl}_x(F_0) \setminus F_0|. \tag{8}$$

Every accepted head belongs to this closure. An empty exact agenda implies a closed fact set containing F_0, hence the same least closure. Since each accepted proposal adds one novel literal, every accepted-only saturating trajectory has the displayed length; rejected proposals only add cost. This specializes the classical closure argument to our proposal cost and stopping rule. Appendix A supplies full derivations.

For witness certification, closure size does not determine the shortest distance. We attempt memoized exhaustive depth-first search over $\mathcal{G}_x$, without heuristic or branch-and-bound pruning. Strict extension makes the graph acyclic, but reachable fact sets can be exponentially many in the number of derivable literals. The audit queries initial states, offline diagnostic paths, and candidate successors. Each distance invocation is capped at 100,000 recursive state visits or two seconds. A separate thirty-second guard limits accumulated production-evaluation time and is checked between calls; Appendix B specifies its scope. Completed searches return exact distances under the assumptions. A search that hits a cap returns no distance, rather than an approximate value or infinity. The evaluator never guides runtime inference.

A policy failure means no correct answer was certified within B proposals. A search cap means the offline distance computation did not finish; it can occur on either a policy success or a policy failure. Let E contain instances with completed initial-state searches, $K = X \setminus E$ those with capped searches, and $O_B = Q_B \cap E$. Then $P_B \cap E \subseteq O_B$. A successful policy trajectory proves membership in Q_B even when the search is capped. For policy failures with capped searches, other evidence is needed. Appendix B relates the original audit, additional distance computations, and standalone anonymous implementation.

What a cap hit leaves unknown. Completed searches identify O_B, while policy successes with capped searches identify $P_B \cap K$. The remaining instances in K may or may not admit a within-budget certifying path, giving the deterministic bounds

$$\frac{|O_B| + |P_B \cap K|}{|X|} \le \mathrm{CA}_B^{\mathrm{ord}\star}(X) \le \frac{|O_B| + |K|}{|X|}. \tag{9}$$

For the gap, uncertainty remains only where both checks are inconclusive: the selected policy fails and the distance search is capped ($K \setminus P_B$). Hence

$$\frac{|O_B \setminus P_B|}{|X|} \le \Delta_B^{\mathrm{ord}}(\pi_0; X) \le \frac{|O_B \setminus P_B| + |K \setminus P_B|}{|X|}. \tag{10}$$

Only policy failures with capped searches contribute to the interval width. These are finite-set logical bounds, not statistical confidence intervals. A verified alternative of at most B proposals proves repairability; a saturation cost above B proves non-repairability for a saturation-only target. Completed shortest-path evaluation resolves other targets. Without such evidence, a policy failure with a capped search remains unclassified. In the original eight-proposal audit, all capped searches concern policy successes, so the interval width is zero.

When is stopping sound? Let $L_x = \mathrm{Cl}_x(F_0)$. For the open-world rule, a sufficient condition is that q and its explicit negation $\bar{q}$ are not both in L_x, and the gold label agrees with closure membership: TRUE for q, FALSE for $\bar{q}$, and UNKNOWN for neither. For the closed-world rule, require YES exactly when $q \in L_x$ and NO otherwise. Every reachable fact set is a subset of L_x, so early witnesses agree with this answer; an empty exact agenda identifies the closure itself.

If both opposite literals were derivable, one correct trajectory would not establish soundness for all orderings. The additional closure checks verify the sufficient condition on the complete primary population; the argument from closure to all reachable states is given in Appendix A.

4 EXPERIMENTS

4.1 DATA, POLICIES, AND SELECTION

Data. We use the `hitachi-nlp/proofwriter_processed_OWA` processed distribution of ProofWriter (Tafjord et al., 2021). The ProofWriter portion of the primary cohort retains 14,860 of its 20,030 depth-5 test questions; the original adapter excluded 5,170 through an additional filter on negative rule premises. The official open-world release treats these premises as explicit negation; the original adapter incorrectly classified their ˜ representation tag as negation as failure. For a controlled comparison of data scopes, we retain this subset and audit the full 20,030 questions with the same selected policy in Section 4.5. Policy selection uses a rulebase-grouped validation partition from the processed training split. PrOntoQA (Saparov & He, 2023) is generated with its public generator, converted, and deduplicated within hop count. A prespecified Secure Hash Algorithm 256-bit digest of each full sample identifier assigns zero modulo ten to validation and the remainder to evaluation. Records preserve no richer template or generated-world identifier; we establish identifier-level disjointness only. Appendix C records generation and split details.

The ProofWriter validation partition has 2,616 questions from 127 rulebases, disjoint from the remaining 24,248 training questions and test. Each PrOntoQA hop starts with 2,000 generated examples; deduplication precedes splitting. Evaluation denominators are reported in Table 1; Appendix C gives all partition counts.

Policies and budgets. All policies receive the same complete agenda. Breadth-first saturation maintains a first-in-first-out queue of enabled rules, inserting rules enabled together by rule index. It selects the earliest inserted rule still available; this queue policy is distinct from offline breadth-first search over states. Query-relevance-first minimizes reverse dependency-graph distance from a candidate head to q or $\bar{q}$ for ProofWriter, and to q alone for PrOntoQA, with rule-index tie breaking. The thirteen prespecified configurations implement twelve distinct rules, since goal-backward-distance is an exact alias of query-relevance-first. Other configurations use deterministic hash-based selection, body length, fan-out, one-step lookahead, or width-two/four receding-horizon beam search. For each setting and $B \in \{1, 2, 4, 8, 16, 32, 64, 128\}$, we select the highest validation certified accuracy, breaking ties by a prespecified configuration order. The primary audit budget is $B = 8$. Appendix D describes policy families and selection; the anonymous implementation supplies full scoring, hashing, and search details. Internal planning is not charged as a proposal: this is not a wall-clock or total-compute-matched comparison.

Setting screen and scope. A prespecified screen considered ProofWriter depths 1, 2, 3, and 5 and PrOntoQA hop counts 1–5 before the repairability audit. It sought settings where legal actions could waste steps, the baseline policies sometimes chose such actions and failed within budget, and exact evaluation was feasible. Appendix C defines the state- and action-level screening rates, their precision requirement, all five thresholds, and all nine outcomes. All thresholds, caps, and $B = 8$ were fixed before screening; neither the screen nor budget choice used R_B or the final ordering gap.

The screen retained ProofWriter depth-5 and PrOntoQA 2–5-hop. It used ProofWriter test subsets and all deduplicated generated PrOntoQA examples, including evaluation examples. These are therefore selected case studies, not an unbiased sample of benchmark settings. Evaluation is held out from *policy selection*, not from setting screening. Claims and uncertainty intervals are conditional on retained settings. All policies, verification, and stopping are deterministic, including fixed-seed hash selection; paired bootstrap intervals resample these evaluated instances, keeping both policies' outcomes paired. They describe variation under this resampling, not uncertainty across generation seeds or benchmark families.

The screen does not force a positive gap: a non-shortest action can delay certification without causing failure at $B = 8$, while saturation can exceed the budget under every ordering.

4.2 THE EIGHT-PROPOSAL CEILING IS ATTAINED

Table 1 reports the primary audit. The selected policies fail on 4,988 of 21,677 instances: 2,730 ProofWriter UNKNOWN answers and 2,258 PrOntoQA NO answers.

Table 1: Primary audit at $B = 8$ across five settings. Certified accuracy equals the ordering ceiling.

Setting	Instances	Failures	Accuracy / ceiling
ProofWriter depth-5	14,860	2,730	81.63%
PrOntoQA			
2-hop	1,536	291	81.05%
3-hop	1,712	446	73.95%
4-hop	1,785	619	65.32%
5-hop	1,784	902	49.44%
All settings	21,677	4,988	76.99%

Table 2: ProofWriter primary cohort ($n = 14{,}860$): success counts, ceiling counts $|Q_B|$, and repairable failures.

B	Selected	Portfolio	Ceiling	Repairable
1	2,868	2,868	[2,868, 2,870]	[0, 2]
2	4,270	4,270	4,270	0
4	7,059	7,112	7,113	54
8	12,130	12,130	12,130	0
16	14,818	14,818	14,818	0

Table 1: failures are UNKNOWN for ProofWriter and NO for PrOntoQA. Selected policies are query-relevance-first for ProofWriter and PrOntoQA 5-hop, and breadth-first saturation for PrOntoQA 2–4-hop. Failure cost ranges and means appear in Table 9. Table 2: the portfolio is the instancewise union of thirteen configurations. Brackets are logical bounds from two unresolved one-proposal cases, distinct from the eight original capped searches later resolved by breadth-first search (Appendix B).

Every residual failure therefore requires saturation under our fixed stopping rules. *This failure composition is the empirical finding;* cost invariance then follows from the model assumptions.

Archived completed rollouts at budget 128 establish that these failures require 9–19 accepted proposals to saturate (median 11; mean 11.687). By Eq. (8), no ordering reduces those costs to eight. Conversely, every selected-policy success supplies a within-budget certifying path. Successful paths and the saturation-cost argument identify $P_8 = Q_8$ on the entire evaluation set and hence $R_8 = \varnothing$. Numerical shortest distances for successful instances are unnecessary for this conclusion. The median budget excess among failures is three proposals (Figure 2(a)).

Exhaustive evaluation provides a separate accounting check. Every PrOntoQA evaluation search completes, and the numbers with $\Phi_x(F_0) \leq 8$ equal the selected-policy successes: 1,245, 1,266, 1,166, and 882 for 2–5 hops. In ProofWriter, $|P_8| = 12{,}130$, $|O_8| = 12{,}122$, and $|K| = 8$. Since $P_8 \cap E \subseteq O_8$ and $|P_8 \cap K| \leq 8$, equality $12{,}130 = 12{,}122 + 8$ forces $P_8 \cap E = O_8$ and $K \subseteq P_8$. Thus all eight capped searches in the original audit concern selected-policy successes. Additional breadth-first search resolves their numerical distances; Appendix B preserves the original cap accounting and reports the refinement. The original 0.054-percentage-point difference between selected accuracy and $|O_8|/|X|$ reflects omitted cap hits, not an ordering gap.

4.3 REPAIRABLE WITNESSES BEYOND A FINITE POLICY PORTFOLIO

The zero gap is budget-specific. On the same 14,860 ProofWriter instances, Table 2 compares the selected policy, the instancewise union of the thirteen configurations, and the full ordering ceiling. At $B = 4$, 54 selected-policy failures are repairable: the finite portfolio finds alternatives for 53, while exact search identifies one additional TRUE witness. All thirteen policies fail on this instance in both the original and anonymous implementations; an independently replayed four-action path succeeds. Appendix E.4 supplies the complete example. This single case demonstrates diagnostic information beyond the finite portfolio, without implying a large empirical advantage for exhaustive search.

Write the relevant initial facts as a, b and the required new literals as u, v, w, q. The successful ordering is

$$a \to u, \qquad \{a, u\} \to v, \qquad v \to w, \qquad \{u, w\} \to q.$$

The proof shares u across both conjunctive rules. Each of u, v, w, q is initially absent and required by the query's dependencies, so four proposals are necessary. The selected policy instead derives z after u. This new literal supports the alternative route $a \to z$, $\{b, z\} \to u$, but u is already known, so this step cannot help the proof and leaves no budget for q. Answer class and closure size identify neither this avoidable step nor the four-step path.

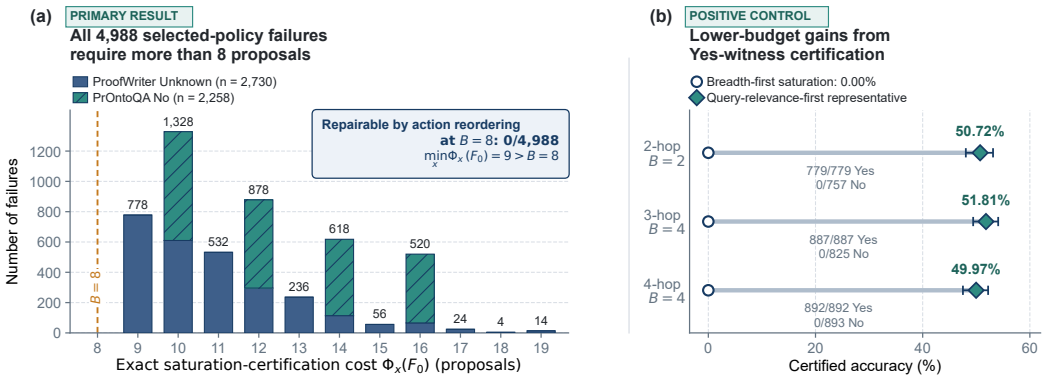


Figure 2: Complementary certification modes in the primary cohort. (a) All 4,988 residual failures at eight proposals have exact saturation costs above budget. (b) At three lower-budget pairs, query-directed ordering certifies every YES instance and no NO instance, while breadth-first certifies neither. Error bars show 95% percentile-bootstrap intervals for the paired accuracy gain from 1,000 instance resamples. Because breadth-first accuracy is zero in every resample, these intervals also locate query-relevance-first accuracy on the plotted axis. Panel (b) compares policies, not exact ceilings.

The two trajectories make the wasted proposal explicit. Each column below records the new literal added by one accepted proposal:

Proposal	1	2	3	4
Selected policy	u	z	v	w
Shortest certifying path	u	v	w	q

Both trajectories obey the same rules. After the common first step, the selected policy spends one of its three remaining proposals on z; the successful path uses all three to derive v, w, and finally q. The positive answer label establishes eventual provability, but deciding whether the four-proposal budget suffices requires examining those dependencies.

4.4 POSITIVE CONTROL: EARLIER WITNESS CERTIFICATION

This control asks whether ordering can help when an early witness is available. We compare PrOntoQA two-hop at $B = 2$, three-hop at $B = 4$, and four-hop at $B = 4$. Query-relevance-first certifies every YES instance and no NO instance; breadth-first saturation certifies neither class. The gains are respectively 50.72, 51.81, and 49.97 percentage points, exactly the YES class proportions (Figure 2(b); counts in Appendix E). For each setting, these two fixed policies tie at the other seven values in the tested budget grid $\{1, 2, 4, 8, 16, 32, 64, 128\}$.

There are respectively 7, 7, and 6 tied validation-maximizing configurations out of thirteen. Query-relevance-first is the representative selected by the prespecified tie order, not a unique maximizer. Across 1,000 paired instance-level bootstrap resamples, the 95% percentile intervals for the gains are [48.05, 53.13], [49.42, 54.09], and [47.51, 52.21] percentage points. Both policies are fixed across evaluation instances. These comparisons establish earlier witness certification. Additional exact audits find that the policies selected separately for each PrOntoQA setting and budget attain the ceilings at $B \in \{1, 2, 4, 8, 16\}$; the policy contrast itself is not that ceiling.

4.5 SCOPE, ORDERING, AND STOPPING SENSITIVITY

Restoring negative premises. The official ProofWriter documentation treats negative premises as explicit negation in the open-world version. For all 20,030 test questions, the official and processed distributions agree on symbolic rules, text, and labels; independent closure evaluation finds no label mismatch or contradictory literal pairs. Normalizing the negative-body tag leaves the adapter otherwise unchanged and preserves all 14,860 original canonical instances. With the original selected policy frozen, the 5,170 added questions yield 4,432 successes and an exact ceiling of 4,438

at $B = 8$. Six witness failures (three TRUE, three FALSE) are repairable; 732 UNKNOWN failures require more proposals. The full split thus has 16,562 successes versus a ceiling of 16,568: the primary zero gap does *not* extend to the full split.

Rule order and generation seeds. Ten deterministic rule-index permutations preserve each instance's logic while changing actual tie order. With selected policies frozen, four permutations each introduce one repairable ProofWriter witness failure at $B = 8$; the other six retain zero gap. All PrOntoQA primary outcomes and the three positive-control comparisons are unchanged. Three additional generation seeds yield 1,066 evaluation instances across 2–5 hops, using the same generator configuration, split rule, and frozen policies. Each seed retains zero primary gap, while the nine lower-budget controls give 41.94–62.64 percentage-point gains, always exactly their YES proportions. These are bounded sensitivity checks, not independent benchmark families; cross-seed canonical overlap is recorded in Appendix E.5.

Changing the negative certificate. In all 3,377 original PrOntoQA NO instances, the explicit opposite of the query is derivable. We therefore test an alternative rule that also certifies NO when that opposite appears, holding policy targets and action choices fixed. At $B = 8$, this raises successes on the 6,817 PrOntoQA instances from 4,559 to 6,320. A separate shortest-distance audit under the alternative stopping rule finds that all 6,817 admit certification within eight proposals. All 497 remaining failures occur at five hops and are now repairable. The intervention changes C_x and hence Φ_x; it does not contradict the zero gap under the original saturation-only protocol. Appendix E.5 reports the paired controls and new-seed results.

5 INTERPRETATION AND LIMITATIONS

Using the diagnosis. The audit separates a policy's behavior from what the fixed system permits. For a repairable failure, an alternative trajectory identifies a concrete ordering change. In the four-proposal ProofWriter example, that change is to derive v after u, rather than spend the second proposal on z. The trajectory establishes a feasible target for improving the policy; whether a learned policy can recover it remains a separate question.

For a saturation failure, the closure count instead identifies the budget needed to complete certification. In the primary eight-proposal audit, the remaining failures require 9–19 proposals regardless of ordering. Increasing the budget to an instance's known saturation cost suffices; retaining the smaller budget requires changing another fixed component. The PrOntoQA stopping-rule comparison illustrates such a change: accepting an earlier negative witness changes which paths can terminate and hence changes the ordering ceiling. When a failed instance has neither a completed distance calculation nor other threshold evidence, the audit retains the bounds in Eqs. (9)–(10).

Scope and computational cost. Exact search can be exponential; one witness beyond the portfolio does not establish scalable or cost-effective auditing. We neither train a ranker nor measure subsequent learning gains. The no-repair result concerns a restricted original cohort, original tie order, stopping protocol, and $B = 8$. The screen used evaluation pools, so these selected case studies do not estimate prevalence across benchmark families. PrOntoQA has identifier-level, not template- or generated-world-level, separation and instance-level resampling. The model excludes incomplete candidate recall, generated actions, learned or stochastic verifier errors, and nonmonotonic revision. Proposal budgets do not normalize internal planning or wall-clock cost.

6 CONCLUSION

The audit asks whether reordering can repair a failed rollout within a fixed system and budget. In the primary cohort with the original rule order and stopping rules, all 4,988 failures at eight proposals require saturation costs of 9–19. These costs and successful trajectories establish that the selected policies attain the ceiling. Lower budgets, broader data scope, and changed rule-index tie order reveal repairable failures; changing the stopping rule changes the ceiling itself. The results support diagnosing what prevents certification before attributing failure to action ordering.

AI USE STATEMENT

We used generative artificial intelligence tools to assist with manuscript writing and language polishing, and with retrieval and discovery of relevant literature. The authors are responsible for the final content of the paper and its accompanying materials.

REPRODUCIBILITY STATEMENT

Appendices A–H document proofs, evaluator caps, data construction and selection, policy families, and auxiliary results. The supplementary archive includes anonymous source code, configuration, aggregate result snapshots, tests, and reproduction instructions. Public benchmark data and the third-party PrOntoQA generator must be obtained separately. The archive distinguishes reproduction code from original result snapshots; static packaging checks do not constitute an independent experimental reproduction.

REFERENCES

François Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs (extended abstract). In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 1–15. Association for Computing Machinery, 1986. doi: 10.1145/6012.15399.

Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An environment for machine learning of higher order logic theorem proving. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 454–463. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/bansal19a.html.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. Natural language deduction through search over statement compositions. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 4871–4883. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-emnlp.358. URL https://aclanthology.org/2022.findings-emnlp.358/.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537.

Jonas Geiping, Sean Michael McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander Fabbri, Wojciech Maciej 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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22017–22031, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1229. URL https://aclanthology.org/2024.emnlp-main.1229/.

Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem

proving. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26337–26349. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/a8901c5e85fb8e1823bbf0f755053672-Paper-Conference.pdf.

Maciej Mikuła, Szymon Tworkowski, Szymon Antoniak, Bartosz Piotrowski, Qiaochu Jiang, Jin Zhou, Christian Szegedy, Łukasz Kuciński, Piotr Miłoś, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39326–39350, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/ab5f5f22e3e09f4424592ffb06840ab0-Paper-Conference.pdf.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.739. URL https://aclanthology.org/2024.acl-long.739/.

Yu Pei, Yongping Du, and Xingnan Jin. FoVer: First-order logic verification for natural language reasoning. *Transactions of the Association for Computational Linguistics*, 13:1340–1359, 2025. doi: 10.1162/tacl.a.41. URL https://aclanthology.org/2025.tacl-1.61/.

Gabriel Poesia, Kanishk Gandhi, Eric Zelikman, and Noah Goodman. Certified deductive reasoning with language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=yXnwrs2Tl6.

Zacchary Sadeddine and Fabian M. Suchanek. Verifying the steps of deductive reasoning chains. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 456–475, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.25. URL https://aclanthology.org/2025.findings-acl.25/.

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

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

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976. doi: 10.1145/321978.321991.

Haiming Wang, Ye Yuan, Zhengying Liu, Jianhao Shen, Yichun Yin, Jing Xiong, Enze Xie, Han Shi, Yujun Li, Lin Li, Jian Yin, Zhenguo Li, and Xiaodan Liang. DT-solver: Automated theorem proving with dynamic-tree sampling guided by proof-level value function. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12632–12646, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.706. URL https://aclanthology.org/2023.acl-long.706/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6984–6994. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/yang19a.html.

Kaiyu Yang, Jia Deng, and Danqi Chen. Generating natural language proofs with verifier-guided search. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 89–105, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.7. URL https://aclanthology.org/2022.emnlp-main.7/.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21573–21612. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4441469427094f8873d0fecb0c4e1cee-Paper-Datasets_and_Benchmarks.pdf.

Hongyu Zhao, Kangrui Wang, Mo Yu, and Hongyuan Mei. Explicit planning helps language models in logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11155–11173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.688. URL https://aclanthology.org/2023.emnlp-main.688/.

These supplementary sections give formal derivations, data construction, setting selection, evaluator accounting, and auxiliary analyses. They do not expand the paper's empirical scope. Each evaluated *action-ordering policy* selects from the same complete agenda of actions accepted by the local verifier; it only reorders that agenda and neither generates nor suppresses candidates. A *ranker-repairable failure* means that some accepted-action ordering succeeds within the same proposal budget, not that such an ordering is learnable from finite data. The *same-budget action-ordering ceiling* is the fraction of instances certifiable by such reorderings while all other system components remain fixed; the *action-ordering gap* is this ceiling minus the selected policy's certified accuracy.

A Formal Details

A.1 Exact Existence on the Accepted-State Graph

The formal results below assume the fixed rule set, complete agenda, unit-cost strict one-literal extension, state-preserving rejection, and sound state-based stopping rule stated in the main paper. Policies may vary their action selection but not these components or the proposal budget.

For an instance x, let F_0 be its initial fact set, $\mathcal{R}_x$ its finite set of grounded Horn-style rules, C_x its state-based stopping rule, and

$$A(F) = \{r \in \mathcal{R}_x : \mathrm{body}(r) \subseteq F,\ \mathrm{head}(r) \notin F\}$$

its exact agenda. Each accepted proposal $r \in A(F)$ has unit proposal cost and produces $F' = F \cup \{\mathrm{head}(r)\}$; a rejected proposal consumes the same unit cost but leaves F unchanged. The complete accepted-state graph $\mathcal{G}_x$ contains every reachable fact set and every accepted transition

$$F \longrightarrow F \cup \{\mathrm{head}(r)\}, \qquad r \in A(F).$$

The implementation stores $s = (F, \delta, A, U)$, where $\delta(r) = |\,\mathrm{body}(r) \setminus F|$ counts unmet premises and $U = \{r : \mathrm{head}(r) \in F\}$ records rules whose heads are already known. These quantities and $A(F)$ are determined by F and the fixed rules; they do not require additional graph nodes for the same fact set. Let $\mathcal{T}_x$ contain the reachable fact sets at which C_x returns the gold label. We write $C_x(F)$ for the stopping result on the runtime state reconstructed from F. The shortest certifying distance is

$$\Phi_x(F) = \min_{T \in \mathcal{T}_x} d_{\mathcal{G}_x}(F, T), \tag{11}$$

with value ∞ if no certifying state is reachable.

Lemma 1 (Existence criterion)**.** *An accepted-action ordering certifies x from F_0 within B proposals exactly when $\Phi_x(F_0) \le B$.*

Proof. Every accepted-only sequence is a path in $\mathcal{G}_x$, so a certifying sequence of length at most B implies that the shortest such path has length at most B. Conversely, a shortest path of length $\Phi_x(F_0) \le B$ can be executed by consecutively proposing its edges. Every edge is verifier-accepted by construction, and the terminal node is in $\mathcal{T}_x$. □

The following proposition restates Proposition 1 and expands its proof.

Proposition 2 (Budget bound)**.** *Under strict one-literal extension, state-preserving rejection, exact agenda maintenance, and a sound state-based stopping rule, every action-selection policy π, including a learned ranker, satisfies*

$$\pi \textit{ correctly certifies } x \textit{ within } B \textit{ proposals} \quad \Longrightarrow \quad \Phi_x(F_0) \le B.$$

Proof. Assume that π certifies after at most B proposals. Delete every rejected proposal from the trajectory. Because rejected proposals preserve the state, the remaining accepted proposals reach the same terminal fact set. Because C_x is state-based, the terminal label is also unchanged. The accepted subsequence is a path in $\mathcal{G}_x$ whose length is no greater than the original proposal count and hence no greater than B. Equation (11) therefore gives $\Phi_x(F_0) \le B$. □

The assumptions are substantive. The proposition does not cover incomplete candidate generation, nonmonotonic state revision, stochastic verifier outcomes, or policies that change the verifier, stopping rule, transition dynamics, or budget. It matches proposal counts, not total computation: a policy may perform internal planning before issuing a proposal.

For an evaluation set X, the same-budget action-ordering ceiling and the gap of a policy π are

$$\mathrm{CA}_B^{\mathrm{ord}\star}(X) = \frac{1}{|X|}\sum_{x\in X}\mathbf{1}[\Phi_x(F_0)\le B]\,,$$
$$\Delta_B^{\mathrm{ord}}(\pi;X) = \mathrm{CA}_B^{\mathrm{ord}\star}(X) - \mathrm{CA}_B(\pi;X). \tag{12}$$

Here $\mathrm{CA}_B(\pi;X)$ is the fraction of X correctly certified by π within B proposals. Proposition 2 makes the gap nonnegative. A successful runtime rollout also proves threshold membership: after rejected proposals are removed, its accepted subsequence is a certifying path and therefore establishes $\Phi_x(F_0)\le B$, even if the resource-capped evaluator does not return the numerical distance.

A.2 Resource-Capped Set Accounting

Let X be an evaluation set, $E\subseteq X$ the instances whose initial-state exact-distance computation completes, and $K = X\setminus E$ the resource-cap hits. For the validation-selected runtime policy π_0, define

$$\begin{aligned} Q_B &= \{x\in X : \Phi_x(F_0)\le B\},\\ P_B &= \{x\in X : \pi_0 \text{ correctly certifies } x \text{ within } B\},\\ O_B &= Q_B\cap E, \qquad R_B = Q_B\setminus P_B. \end{aligned} \tag{13}$$

Proposition 2 gives

$$P_B\subseteq Q_B, \qquad P_B\cap E\subseteq O_B. \tag{14}$$

Thus R_B contains exactly the selected-policy failures for which a same-budget accepted-action reordering exists, and $\Delta_B^{\mathrm{ord}}(\pi_0;X) = |R_B|/|X|$. On E, membership is resolved as $R_B\cap E = O_B\setminus P_B$. A cap hit is not assigned an infinite distance and is not evidence of non-repairability. Nevertheless, a cap-hit selected-policy success belongs to Q_B by Eq. (14). Consequently,

$$\frac{|O_B|+|P_B\cap K|}{|X|} \le \mathrm{CA}_B^{\mathrm{ord}\star}(X) \le \frac{|O_B|+|K|}{|X|}. \tag{15}$$

Proposition 3 (Cap accounting)**.** *If $|P_B| = |O_B|+|K|$, then $P_B\cap E = O_B$, $K\subseteq P_B$, and $Q_B = O_B\uplus K = P_B$. Consequently, $R_B=\varnothing$ and the full action-ordering gap is zero.*

Proof. The disjoint decomposition $X = E\uplus K$ and Eq. (14) imply

$$|P_B| = |P_B\cap E| + |P_B\cap K| \le |O_B| + |K|.$$

Equality of the two endpoints forces equality in both component bounds. Hence $|P_B\cap E| = |O_B|$ and Eq. (14) gives $P_B\cap E = O_B$; also $|P_B\cap K| = |K|$, so $K\subseteq P_B$. Because $P_B\subseteq Q_B$, every member of K then belongs to Q_B; because $O_B = Q_B\cap E$, the disjoint decomposition gives $Q_B = O_B\uplus K = P_B$ and $R_B=\varnothing$. □

A.3 Order-Invariant Cost of Saturation

Let $\mathrm{Cl}_x(F_0)$ denote the least fixed point obtained by repeatedly firing all grounded rules whose premises are present. Explicit negation is treated as a literal; negation as failure is outside the scope.

Lemma 2 (Saturation invariance)**.** *Under exact agenda maintenance and strict one-literal state extension, every accepted-only trajectory continued until the agenda is empty terminates at $\mathrm{Cl}_x(F_0)$ and uses exactly*

$$\big|\mathrm{Cl}_x(F_0)\setminus F_0\big|$$

accepted proposals. If the stopping rule returns the gold label at the saturated fixed point and returns undecided at every earlier reachable state, this quantity equals $\Phi_x(F_0)$.

Proof. Every accepted rule head belongs to the least fixed point, so every intermediate fact set is a subset of $\mathrm{Cl}_x(F_0)$. If a trajectory ends with an empty exact agenda, its terminal fact set is closed under all rules and contains F_0; minimality of the least fixed point makes it equal to $\mathrm{Cl}_x(F_0)$. Each accepted proposal adds exactly one new literal, so every saturating trajectory has the stated length, independent of ordering. Under the stated stopping condition, every certifying path must reach this

fixed point, while every saturating trajectory reaches it and is certifying. The common length is therefore the shortest certifying distance. □

This is a direct counting consequence of monotone closure and strict one-literal extension. The empirical finding at the primary budget is that every selected-policy failure belongs to a saturation-certified class; that composition is not implied by the lemma. Under our fixed stopping rules, a ProofWriter UNKNOWN answer is returned only after agenda exhaustion, and a PrOntoQA NO answer is returned only after saturation without deriving the query. Ordering can accelerate *witness certification*, which stops after deriving the query or its explicit negation, but it cannot reduce the number of novel literals required for *saturation certification*. The complete accepted-state graph defines the criterion for arbitrary certifying terminal sets. Saturation-only targets are a special case: once certification is known to occur only at agenda exhaustion, Lemma 2 identifies the exact distance from the order-invariant closure size without enumerating alternative orderings.

A.4 A SUFFICIENT CONDITION FOR STOPPING-RULE SOUNDNESS

Stopping-rule soundness is a semantic assumption, distinct from local step legality and from checking one returned path. Let $L_x = \mathrm{Cl}_x(F_0)$ and let $\bar{q}$ denote the explicit negation of the query. For the open-world rule, a sufficient condition is that q and $\bar{q}$ are not both in L_x, and that the gold label is TRUE when $q \in L_x$, FALSE when $\bar{q} \in L_x$, and UNKNOWN when neither belongs to L_x. For the closed-world rule, it suffices that the gold label is YES exactly when $q \in L_x$ and NO otherwise.

To see why these conditions suffice, every reachable fact set is a subset of L_x. An early derived witness therefore agrees with the gold label; the exclusion of simultaneous open-world witnesses prevents an ordering from reaching a conflicting label first. An empty exact agenda identifies L_x itself and hence the correct saturation label. This gives a sufficient semantic condition. Our additional independent closure audit verifies its premises on all 21,677 primary instances, with no label mismatch or query/opposite conflict. In particular, matching the gold label on one returned path, or on one saturated state without checking simultaneous witnesses, does not establish open-world soundness over all reachable states.

B EXACT EVALUATOR AND RESOURCE CAPS

The *original evaluator* produced the archived audit results. The *anonymous implementation* is the standalone reproduction code supplied with this paper. These are software implementations, distinct from the audit records in Table 4. In particular, an originally capped search can later be resolved without changing its archived status.

For $r \in A(F)$, write $F_r = F \cup \{\mathrm{head}(r)\}$. The offline evaluator implements the recurrence

$$\Phi_x(F) = \begin{cases} 0, & F \in \mathcal{T}_x, \\ \infty, & A(F) = \varnothing,\ F \notin \mathcal{T}_x, \\ 1 + \min\limits_{r \in A(F)} \Phi_x(F_r), & \text{otherwise.} \end{cases} \tag{16}$$

It uses memoized exhaustive depth-first evaluation of the finite directed acyclic graph. It performs no heuristic pruning or branch-and-bound search. Within a completed invocation using a shared memoization cache, each fact set is expanded at most once; recursive state visits may also include cache hits. Separate calls with cleared caches can expand the same fact set again. If m literals can be newly derived, the number of reachable fact sets can be exponential in m, which is why completed and capped computations must be distinguished.

The audit requests distances and distance-decreasing action sets at initial, intermediate, and candidate-successor states. An action-set request can invoke the distance solver separately for the current state and several successors. Each underlying distance invocation stops after 100,000 recursive state visits or 2 seconds. A separate guard checks accumulated production-call time between external requests, with a fresh 30-second allowance for each instance in the trajectory-collection and metric-computation phases. It excludes separate cold-cache timing probes and does not preempt an ongoing request. Thus 30 seconds is not a bound on the complete multi-phase audit of an instance. A completed invocation returns an exact value. A cap hit returns no numerical distance; its threshold status remains unresolved unless a successful runtime rollout or separate exact analysis resolves

Table 3: Original-audit set accounting at the primary eight-proposal budget. E contains completed initial-state exact-distance searches, K contains resource-cap hits, P_8 contains selected-policy successes, O_8 contains completed distances at most eight, Q_8 is the full within-budget set, and $R_8 = Q_8 \setminus P_8$ contains failures repairable by action reordering. All eight ProofWriter cap hits reached the 100,000-state-visit cap; none reached the time cap. Their selected-policy successes prove membership in Q_8 without relying on their numerical distances. Subsequent breadth-first refinement resolves these eight distances; the original counts are preserved here.

Setting	Validation-selected ordering	$\lvert X\rvert$	$\lvert E\rvert$	$\lvert K\rvert$
ProofWriter depth-5	query-relevance-first	14,860	14,852	8
PrOntoQA 2-hop	breadth-first saturation	1,536	1,536	0
PrOntoQA 3-hop	breadth-first saturation	1,712	1,712	0
PrOntoQA 4-hop	breadth-first saturation	1,785	1,785	0
PrOntoQA 5-hop	query-relevance-first	1,784	1,784	0
Total		21,677	21,669	8

Setting	$\lvert P_8\rvert$	$\lvert O_8\rvert$	$\lvert Q_8\rvert$	$\lvert R_8\rvert$
ProofWriter depth-5	12,130	12,122	12,130	0
PrOntoQA 2-hop	1,245	1,245	1,245	0
PrOntoQA 3-hop	1,266	1,266	1,266	0
PrOntoQA 4-hop	1,166	1,166	1,166	0
PrOntoQA 5-hop	882	882	882	0
Total	16,689	16,681	16,689	0

whether certification fits the budget. The evaluator is analysis-only and supplies neither actions nor feedback to runtime inference.

The original evaluator treats every non-undecided stopping label as terminal, which agrees with $\mathcal{T}_x$ under the soundness assumption. The anonymous implementation additionally compares the terminal label with the gold answer: a matching label has distance zero, while an incorrect terminal is treated as a dead end with infinite distance and is not expanded. Both conventions agree under the stated soundness assumption. This label check belongs to offline analysis and is not a runtime policy input. Paths extracted offline from the computed distances match the gold labels on all 21,669 evaluator-resolved evaluation instances. This is a path-level consistency check, not an exhaustive verification of stopping-rule soundness at every reachable state.

For each PrOntoQA setting, Eq. (14) and $|P_8| = |O_8|$ imply $P_8 = O_8$. For ProofWriter,

$$12{,}130 = 12{,}122 + 8 = |O_8| + |K|,$$

so Proposition 3 gives $P_8 \cap E = O_8$, $K \subseteq P_8$, and $Q_8 = O_8 \uplus K = P_8$. The eight instances with originally unresolved numerical distances are therefore selected-policy successes: their successful trajectories already prove certification possible within eight proposals. All 4,988 selected-policy failures belong to E and have exact distance greater than eight. The same-budget action-ordering ceiling is thus 16,689/21,677=76.99% overall, and the ordering gap is zero in every setting and overall. The smaller ratio $|O_8|/|X|$ is only the proportion with completed numerical distances at most eight; it is not the full ceiling.

Additional independent checks and refinement. A separate forward-closure implementation verifies the sufficient stopping condition for all 21,677 canonical primary instances. Replaying all 53 archived four-proposal repair witnesses reproduces their failed selected rollout and successful alternative in both implementations, with no outcome disagreement. All eight original cap-hit selected successes also reproduce. Additional breadth-first search, capped at one million discovered states and ten seconds per instance, resolves those eight initial distances and independently checks every returned path. Table 2 combines this refinement with the additional primary-cohort audit. That audit has two other ProofWriter instances whose one-proposal membership remains unresolved; neither belongs to the eight original capped searches. The original counts in Table 3 remain a record of the original computation, not the number still unresolved after refinement. Membership is exact for the

primary ProofWriter cohort at budgets 2, 4, 8, and 16, and for all four PrOntoQA cohorts at budgets 1, 2, 4, 8, and 16.

Table 4: Relation among the audit records. A capped distance search and a policy failure are different events. Counts from separate computations are not silently merged.

Audit record	Evaluation population	Evidence used in the reported result
Original audit	Primary cohort: 21,677	Table 3 preserves the eight capped initial-state searches. All eight are policy successes at $B = 8$.
Additional audit and refinement	Same primary cohort	Table 2 uses the additional audit and breadth-first refinement of those eight cases. Two different cases remain unresolved at $B = 1$.
Separate full-scope audit	ProofWriter: 20,030	Appendix E.5 reports this computation's own bounds, without importing the primary-cohort refinement.

C Benchmarks, Splits, and Setting Selection

C.1 Data Construction

ProofWriter (Tafjord et al., 2021) is obtained from the Hitachi-maintained processed open-world distribution, `hitachi-nlp/proofwriter_processed_OWA`.[1] Its depth-5 test input contains 20,030 questions. The adapter applies an additional representation filter: a rulebase is rejected if a parsed rule contains an atom tagged ˜. This excludes 5,170 questions and retains 14,860; no other conversion error was recorded. The original implementation names this category "negation as failure," which is incorrect for the open-world release. The official `V2020.12.3` README specifies that negative rule conditions use hard negation in the open-world version, while its representation documentation uses ˜ for negative rule premises and restricts the negation-as-failure interpretation to the closed-world version. The source paper's Appendix A.2 likewise explains this semantic distinction (Tafjord et al., 2021).[2] The excluded questions therefore remain within the signed-literal monotone theory. The restriction is an adapter artifact, not a necessary theoretical scope condition.

An additional source audit aligns all 20,030 questions by normalized identifier. Symbolic representations, natural-language content, and gold labels agree exactly between the processed input and official test file. Proof annotations differ for 9,848 questions; no proof-field equality or independent source-proof verification is claimed. Under explicit negation, independent grounding and closure agree with every gold label and contain no opposite-literal conflicts. Interpreting the same negative premises as negation as failure instead changes 1,966 question-level closures and disagrees with 94 labels, including 13 query conflicts. These empirical checks supplement the documented semantics rather than define it.

For the scope extension, only parsed atom polarity tags in rule representations are normalized from ˜ to the adapter's explicit-negative tag. Original prose, labels, and proof fields are preserved. All 14,860 retained canonical objects match their original counterparts field by field. We freeze the original validation-selected policies rather than reselect on the expanded test split; Section E.5 reports the additional 5,170 questions separately.

The validation partition is drawn from the analogously filtered processed training split. A Secure Hash Algorithm 256-bit digest of each rulebase identifier assigns complete rulebase groups to partitions: 2,616 questions from 127 rulebases form validation, and no rulebase group overlaps the remaining 24,248 training questions or the processed test split.

[1] `https://huggingface.co/datasets/hitachi-nlp/proofwriter_processed_OWA`

[2] Official archive and included README: `https://aristo-data-public.s3.amazonaws.com/proofwriter/proofwriter-dataset-V2020.12.3.zip`.

Table 5: Separate full-pool evaluator profiling. An external evaluator call requests a distance or distance-decreasing action set at one state and may reuse cached values; it is not one recursive state visit. Instance time sums production-call time across audit phases, and whole-audit time includes the remaining workflow. The PrOntoQA pools combine validation and policy-selection-held-out evaluation instances, so these timings are not evaluation-only summaries for Table 3. The profiling records wall-clock time, not memory, branching factor, or mean explored-state count; no claim is made for those unrecorded quantities. These calls differ from the fresh-cache screening probes in Table 8.

Profiling pool	Instances	External call 95th percentile (ms)	Cumulative per-instance 95th percentile (ms)	Whole audit (s)
ProofWriter depth-5	14,860	0.039	42.01	228.58
PrOntoQA 2-hop	1,715	0.022	22.42	72.31
PrOntoQA 3-hop	1,914	0.023	27.10	122.63
PrOntoQA 4-hop	1,993	0.023	27.35	178.95
PrOntoQA 5-hop	2,000	0.024	27.88	283.84

Table 6: PrOntoQA generation, within-hop deduplication, and retained-setting partitions. A Secure Hash Algorithm 256-bit digest of the full sample identifier assigns zero modulo ten to validation and the remainder to the policy-selection-held-out evaluation partition. The screen below uses each full deduplicated pool, including both partitions. The canonical records do not preserve template or generated-world identifiers, so the split guarantees identifier-level disjointness only.

PrOntoQA	Generated	Deduplicated	Validation	Evaluation
1-hop	2,000	1,384	–	–
2-hop	2,000	1,715	179	1,536
3-hop	2,000	1,914	202	1,712
4-hop	2,000	1,993	208	1,785
5-hop	2,000	2,000	216	1,784

PrOntoQA (Saparov & He, 2023) is generated from a fixed public generator revision (`066f73fc70bf`). For each hop count from one through five, the generator targets 2,000 examples with seed 62,471,893, a fictional ontology, no optional extra distractor rule, no adjective-based surface realization, and at most 40,000 attempts. Disabling the extra distractor does not remove query-irrelevant consequences already present in the ontology; the adjective option controls wording, not the presence of unary predicates. All 10,000 requested examples are produced and converted without unsupported rules or conversion errors. Deduplication is performed within each hop count using the initial facts, rules, query, any stored explicit-opposite query, and answer in the canonical representation. This representation assigns integer indices to ground literals within each instance. The key compares these indices but omits their mapping to raw signed atoms. ProofWriter stores the opposite query; the original PrOntoQA representation does not, so that part of its deduplication key is always empty. The opposite reconstructed for the stopping-rule experiment is separate and is not added to this key. It retains 1,384, 1,715, 1,914, 1,993, and 2,000 examples for one through five hops, respectively. There are 87 duplicate canonical records across different hop strata; they are retained, while no reported setting denominator contains a within-setting duplicate. The count of 87 concerns the full pools before partitioning, not evaluation-only overlap. The total of 21,677 sums setting-specific evaluation counts and is not globally deduplicated across settings.

C.2 Prespecified Setting Screen

Before the repairability audit, nine setting pools were screened at $B = 8$. The evaluation partitions were held out from policy selection but used by this prespecified screen. Consequently, downstream empirical claims and confidence intervals are conditional on the five retained settings rather than unbiased estimates over a population of benchmark settings. Neither the screen nor the choice of $B = 8$ used the final ordering gap or repairability residual. For each resolved instance, the screen

Table 7: Behavioral criteria for all settings considered by the prespecified screen. The state column measures whether an accepted action outside $A_x^+(F)$ exists; the selection column measures how often a policy chooses one. The last column lists failures among conditions (i)–(iii).

Setting	Pool n	States admitting an action outside $A_x^+(F)$ (%)	Maximum observed selection rate (%)
ProofWriter depth-1	14,754	7.51	5.65
ProofWriter depth-2	14,080	13.24	10.75
ProofWriter depth-3	13,034	19.59	14.99
ProofWriter depth-5	14,860	26.13	24.72
PrOntoQA 1-hop	1,384	13.09	9.81
PrOntoQA 2-hop	1,715	19.75	16.64
PrOntoQA 3-hop	1,914	24.21	21.07
PrOntoQA 4-hop	1,993	26.35	23.42
PrOntoQA 5-hop	2,000	28.12	24.80

Setting	Better certified accuracy (%)	Failed behavioral conditions
ProofWriter depth-1	99.80	i, ii, iii
ProofWriter depth-2	99.38	i, iii
ProofWriter depth-3	97.94	iii
ProofWriter depth-5	81.63	none (retained)
PrOntoQA 1-hop	85.04	i, ii
PrOntoQA 2-hop	80.93	none (retained)
PrOntoQA 3-hop	74.03	none (retained)
PrOntoQA 4-hop	65.08	none (retained)
PrOntoQA 5-hop	49.60	none (retained)

follows one shortest path, breaking action ties by rule index, and retains the states with positive distance. It pools these states across instances. An accepted action outside $A_x^+(F)$ does not decrease the shortest distance; this is the specific kind of alternative measured here. A setting was retained only when all five prespecified conditions held:

(i) At least 15% of the pooled positive-distance states admitted an accepted action outside $A_x^+(F)$.

(ii) The larger observed rate at which query-relevance-first or breadth-first saturation selected such an action was at least 10%.

(iii) The better of those two certified accuracies was at most 90%.

(iv) The 95th-percentile exact-evaluation time for a state queried with a fresh cache was at most 200 milliseconds.

(v) The 95% percentile-bootstrap interval for condition (i), using 1,000 resamples and seed 62,471,893, had half-width at most 2.5 percentage points.

D Action-Ordering Policies and Selection

The study evaluates 13 prespecified configurations representing 12 distinct action-selection rules:

1. hash-based random valid-action selection with seed zero;
2. breadth-first saturation, selecting the earliest inserted available rule in a first-in-first-out activation queue;
3. query-relevance-first, minimizing reverse dependency-graph distance from the rule head to q or $\bar{q}$ for ProofWriter, and to q alone for PrOntoQA;
4. shortest-body-first, minimizing the number of rule premises;

Table 8: Feasibility and precision criteria for the prespecified setting screen. Conditions (i)–(v) and $B = 8$ were fixed before screening. The screen used the evaluation pools to select settings, not policies; all subsequent claims and uncertainty intervals are therefore conditional on the retained settings rather than unbiased estimates over a population of settings. These cold timings are screening diagnostics and are distinct from the full-pool profiling in Table 5. Every setting passes conditions (iv) and (v).

Setting	Cold 95th percentile (ms)	Interval half-width (percentage points)
ProofWriter depth-1	0.029	0.43
ProofWriter depth-2	0.031	0.54
ProofWriter depth-3	0.032	0.60
ProofWriter depth-5	0.040	0.72
PrOntoQA 1-hop	0.020	1.28
PrOntoQA 2-hop	0.020	1.62
PrOntoQA 3-hop	0.020	1.70
PrOntoQA 4-hop	0.021	1.81
PrOntoQA 5-hop	0.022	1.82

5. goal-backward-distance, which is behaviorally identical to query-relevance-first in the grounded representation used here;
6. maximum-fan-out and minimum-fan-out, where fan-out counts rules that use the candidate head as a premise;
7. one-step lookahead with either query relevance or breadth-first inner scoring; and
8. width-2 and width-4 receding-horizon beam search with either query relevance or breadth-first inner scoring.

Rule-index order resolves action-score ties; stable creation order additionally resolves beam ties. The candidate configuration order resolves equal validation certified accuracy. Throughout, *validation-selected* denotes the first configuration under this prespecified order among all validation maximizers; it is a tie-broken representative, not necessarily a unique maximizer. Selection is repeated for each setting and proposal budget in $\{1, 2, 4, 8, 16, 32, 64, 128\}$. For breadth-first saturation, newly enabled rules receive increasing queue positions, ordered by rule index when enabled together. Queue positions are maintained along the executed trajectory, not by breadth-first search over the complete state graph. For query relevance, PrOntoQA's reconstructed opposite remains excluded from scoring even in the alternative stopping experiment; only the stopping rule uses it.

One-step lookahead simulates each available action and prioritizes successors that satisfy the stopping rule. Otherwise it minimizes the best query-distance or queue-position score among the successor's available actions, breaking ties by the candidate action's rule index. Beam search uses these same nonterminal successor scores, accumulated along each internal path. After t proposals it plans at most $B - t$ steps ahead, retaining the best two or four nonterminal candidates at each depth. Terminal candidates are stored separately and preferred at shallower depth. The policy executes only the first action of the chosen internal path and then replans. Internal score ties use rule-index and stable creation order.

Every policy receives the same complete grounded agenda. The external budget counts issued proposals. Internal successor evaluations by one-step lookahead and internal beam expansions are not charged as proposals, so the comparison is proposal-matched rather than total-compute-matched. The exact-distance evaluator is not one of the candidate policies.

E Additional Experimental Results

E.1 Failure Distances and a Saturation Case

At $B = 8$, the selected policies leave 4,988 failures. All 2,730 ProofWriter failures are open-world UNKNOWN answers, and all 2,258 PrOntoQA failures are closed-world NO answers. Table 9 reports

Table 9: Exact saturation-certification costs for selected-policy failures at $B = 8$. The overall median is 11 proposals; the median excess over the primary budget is three proposals, and the maximum excess is 11. These are distances for the stated saturation targets, not a multi-budget repairability audit.

Setting	Failures	Label	Range	Mean
ProofWriter depth-5	2,730	Unknown	9–19	10.92
PrOntoQA 2-hop	291	No	10–16	12.67
PrOntoQA 3-hop	446	No	10–16	12.66
PrOntoQA 4-hop	619	No	10–16	12.67
PrOntoQA 5-hop	902	No	10–16	12.53
All	4,988		9–19	11.69

Table 10: Pooled exact saturation-certification costs for the 4,988 selected-policy failures at $B = 8$ across the five retained settings. The counts sum to 4,988 and yield range 9–19, median 11, and mean 11.687. The pooled set contains 2,730 ProofWriter UNKNOWN and 2,258 PrOntoQA NO instances.

Cost $\Phi_x(F_0)$	Failures	Cost $\Phi_x(F_0)$	Failures
9	778	15	56
10	1,328	16	520
11	532	17	24
12	878	18	4
13	236	19	14
14	618		

their exact saturation-certification costs. The archived completion counts at $B = 128$, together with Lemma 2, determine these exact distances for targets whose certification requires saturation. All failures complete at $B = 128$. Thus the primary no-repair result follows from the saturation-cost argument: each selected-policy failure has an order-invariant cost greater than eight, while each selected-policy success supplies an explicit within-budget certifying path. Together these identify the full ordering ceiling without requiring numerical shortest distances for the successful instances. The exhaustive-search counts in Table 3 provide a separate accounting check. General shortest-path analysis remains relevant to witness-certified instances, for which closure size alone does not determine the earliest possible certificate.

The PrOntoQA 4-hop Sally example introduced in Section 1 illustrates the mechanism. Its initial fact is `dumpus(Sally)`, the query is `large(Sally)`, and the gold answer is NO. Saturation adds 12 new ground literals, written below by predicate with the shared argument Sally omitted: `yumpus`, `tumpus`, `numpus`, `amenable`, `not luminous`, `not large`, `impus`, `wumpus`, `not orange`, `zumpus`, `sweet`, and `transparent`. Both query-relevance-first and breadth-first saturation fail at eight proposals and certify NO after 12 proposals when given a larger budget. Even though the explicit negative literal $\neg$`large(Sally)` is derivable, our fixed closed-world stopping rule certifies NO only after agenda exhaustion. By Lemma 2, changing the order cannot reduce this 12-proposal saturation cost. Accepting the explicit opposite as an earlier negative certificate would change the stopping protocol. The twelve-step cost follows from the fixed stopping rule used here. Appendix E.5 quantifies the effect of the alternative rule on the full PrOntoQA evaluation cohort.

E.2 POSITIVE CONTROL: EARLIER WITNESS CERTIFICATION

The lower-budget gains have a specific class-conditional mechanism. Query-relevance-first certifies every positive YES instance in the three highlighted comparisons and no NO instance. Breadth-first saturation certifies neither class at those budgets. These gains therefore come from reaching positive witnesses earlier; they do not reduce saturation-certification cost, which explains why the UNKNOWN and NO residual failures at $B = 8$ remain unrepairable by action ordering.

Table 11: Class-conditional decomposition of the three lower-budget gains. Intervals concern the accuracy gain and use 1,000 paired instance resamples. Each comparison has a reproducible random stream derived from base seed 1,729 and its benchmark, setting, budget, and policy pair; the final seeds are supplied with the reproduction code. Query-relevance-first is the representative chosen by the prespecified configuration-order tie break; it is not the unique validation maximizer at these points.

Setting / B	Class size Yes / No	Breadth-first successes Yes / No	Query-relevance successes Yes / No
PrOntoQA 2-hop / 2	779 / 757	0 / 0	779 / 0
PrOntoQA 3-hop / 4	887 / 825	0 / 0	887 / 0
PrOntoQA 4-hop / 4	892 / 893	0 / 0	892 / 0

Setting / B	Gain [95% interval] (percentage points)	Tied validation configurations
PrOntoQA 2-hop / 2	50.72 [48.05, 53.13]	7
PrOntoQA 3-hop / 4	51.81 [49.42, 54.09]	7
PrOntoQA 4-hop / 4	49.97 [47.51, 52.21]	6

The 2-hop Alex example in Section 1 begins with `yumpus(Alex)` and asks whether `happy(Alex)` holds. The certifying chain is

$$\texttt{yumpus}(x) \rightarrow \texttt{impus}(x) \rightarrow \texttt{happy}(x).$$

The rule $\texttt{yumpus}(x) \rightarrow \texttt{hot}(x)$ is irrelevant to the query. The rules deriving `impus(Alex)` and `hot(Alex)` enter the initial queue in that order. Both policies first derive `impus(Alex)`, which enables the rule deriving `happy(Alex)`. The rule deriving `hot(Alex)` was already queued, so breadth-first saturation selects it next and cannot certify within two proposals. Query-relevance-first instead selects the newly enabled query rule and certifies YES on its second proposal.

At 2-hop/2 and 3-hop/4, the seven tied configurations are query-relevance-first, its goal-backward-distance alias, one-step query relevance lookahead, both beam widths with query relevance, and both beam widths with breadth-first inner scoring. At 4-hop/4 the same set ties except for width-2 beam search with breadth-first inner scoring. The result therefore supports the usefulness of query-directed ordering at these budgets, not the unique optimality of a particular implementation.

E.3 Selection Agreement and Descriptive Budget Summaries

As a retrospective check, applying the same tie order to evaluation accuracy selects the same tie-broken representative as validation accuracy in all 40 setting–budget pairs; this check was not used for policy selection. For the three fixed query-relevance-first versus breadth-first comparisons in Table 11, we also summarize the chosen budget grid by giving each of its eight budget values equal weight. The mean certified-accuracy gain over all eight budgets is 6.34, 6.48, and 6.25 percentage points. Each fixed-policy pair has equal certified accuracy at the other seven budgets in the corresponding eight-point grid. Each mean is consequently the single nonzero gain divided by eight. These averages describe the chosen budget grid. They do not provide additional evidence of robustness to new data or perturbations, because each reuses the same single nonzero comparison.

Agreement between validation-selected and retrospectively evaluation-selected configurations does not imply equality with breadth-first saturation. Table 12 separates these quantities for five additional setting–budget comparisons in the archived policy matrix. ProofWriter at four proposals and PrOntoQA five-hop at eight proposals exhibit substantial baseline differences; the other three rows are ties.

These policy curves alone do not identify the full ordering ceiling. Table 2 gives the additional exact-distance evidence for ProofWriter; Appendix B states the corresponding PrOntoQA budget coverage. The finite portfolio remains a lower bound and misses one four-proposal witness.

Table 12: Additional archived policy comparisons. The selected-policy column reports the validation-selected configuration; the baseline is breadth-first saturation. A separate check compares which configuration is chosen using validation accuracy versus evaluation accuracy across five settings and eight budget values; it does not compare either selection with this baseline.

Setting / B	Breadth-first accuracy (%)	Selected-policy accuracy (%)
ProofWriter depth-5 / 4	25.73	47.50
PrOntoQA 2-hop / 4	62.30	62.30
PrOntoQA 3-hop / 8	73.95	73.95
PrOntoQA 4-hop / 8	65.32	65.32
PrOntoQA 5-hop / 8	0.00	49.44

Table 13: ProofWriter scope extension at $B = 8$ with the original selected policy frozen. Counts refer to successes or instancewise ceiling membership. The added witness failures make the full-split gap nonzero.

Population	Instances	Selected	Ceiling count	Repairable
Original retained	14,860	12,130	12,130	0
Added negative premises	5,170	4,432	4,438	6
Full depth-5 test	20,030	16,562	16,568	6

E.4 A Witness Missed by All Tested Policies

The additional ProofWriter example has gold answer TRUE. We use the same literal names as Section 4.3: a, b are initial facts, u, v, w are intermediate literals, z lies on the unused alternative route, and q is the query. These names abbreviate ground literals of this one instance. The successful four-proposal path is

$$a \to u, \qquad \{a, u\} \to v, \qquad v \to w, \qquad \{u, w\} \to q.$$

Each step is enabled by the initial facts or earlier steps, adds exactly one new literal, and the fourth derives the query. In the complete stored rule set, these dependencies also require at least four proposals: q requires w, which requires v, which requires the initially absent u.

Thus the path has exact length four. The supplementary evidence preserves all 30 grounded rules and the query and its explicit opposite, not only this subproof.

The query-relevance-first rollout derives u, z, v, w, in that order, using $a \to z$ for its second proposal. This new literal satisfies the premises of the alternative rule $\{b, z\} \to u$, whose head u is already known. The accepted step adds z; it does not rederive u. The query remains absent after its fourth proposal. All thirteen configurations fail within four proposals in both implementations, including the goal-backward-distance alias and the lookahead/beam configurations. Separately, all 53 original portfolio-repairable witness records reproduce. Their union with this additional witness gives the 54 exact repairable failures in Table 2. The example distinguishes an existence certificate from failure of a finite collection of policies.

E.5 Additional Scope and Protocol Sensitivity

These follow-up analyses keep the original primary cohort and all original validation selections intact. They assess specific boundary conditions rather than retroactively changing the setting screen.

ProofWriter negative-premise extension. The additional 5,170 questions comprise 1,386 TRUE, 1,386 FALSE, and 2,398 UNKNOWN labels. Source normalization changes 315 rule representations and preserves original natural-language and proof fields. Canonical identity checks cover every retained original question. The original query-relevance-first selection is frozen at both four and eight proposals; no expanded-validation reselection or test-based selection occurs.

Table 14: New-seed positive controls. Entries show query-relevance-first successes divided by evaluation size, followed by gain in percentage points. Breadth-first saturation has zero successes in all nine comparisons.

Seed	2-hop, $B = 2$	3-hop, $B = 4$	4-hop, $B = 4$
62,471,894	38/86; 44.19	52/87; 59.77	57/91; 62.64
62,471,895	50/86; 58.14	43/88; 48.86	38/88; 43.18
62,471,896	39/93; 41.94	39/91; 42.86	49/88; 55.68

All added-scope searches complete. At four proposals, the added group has 2,288 selected successes versus ceiling 2,302, giving 14 repairable failures. At eight, its six repairable failures are evenly divided between TRUE and FALSE; the other 732 failures are UNKNOWN saturation cases above budget. For comparison, breadth-first saturation certifies 1,507 and 4,112 added questions at four and eight proposals, respectively. The separate full-scope run records ten capped numerical distances in the old cohort, but success certificates resolve every eight-proposal membership. Using that computation's evidence alone, the four-proposal full-scope repairability bounds are [68,70]. This interval preserves the separate full-scope computation; it does not combine the refined 54 old-cohort failures in Table 2 with the 14 added-scope failures. Table 4 distinguishes these records.

Effective rule-index perturbations. For each of the 21,677 primary instances, seeds 0–9 deterministically permute the rule indices and rule positions together, preserving heads, premises, initial facts, and query literals. This changes the actual policy tie order in the normalized rule representation used by the policies. Each permuted rule receives its new positional index; the policies therefore see the changed order when breaking ties. Shuffling raw input lines alone could be undone by normalization. The original and permuted samples use identical policies, budgets, and stopping rules, without reselection. All ten permutations retain PrOntoQA primary successes 1,245, 1,266, 1,166, and 882, and preserve the three highlighted positive-control contrasts. ProofWriter successes at eight proposals are 12,129 for seeds 3, 4, 6, and 7, and 12,130 for the other six seeds. Each loss is one witness that was originally certified within eight, so it is repairable under the unchanged graph. The strict zero is therefore sensitive to tie order, although the observed effect is one instance per affected permutation. Permutation seeds are repeated interventions on the same examples, not independent data samples.

New generation seeds. We use seeds 62,471,894–62,471,896, target 100 generated examples per seed and hop (2–5), and keep the fictional ontology, no-distractor, no-adjective settings and generator revision fixed. Deduplication uses the original canonical key and precedes identifier-hash splitting. The twelve groups retain 1,066 evaluation instances; validation groups are recorded but not used to reselect policies or rescreen settings. At $B = 8$, frozen selected policies reach the exact ceiling in each group. Paired controls at the original three lower budgets are reported in Table 14. In every row, breadth-first saturation certifies neither answer class, and query-relevance-first certifies all YES and no NO instances. The gain is consequently the observed class proportion, not a seed-invariant fifty-point constant.

Table 15 gives the partition sizes for each seed and hop count. Some original-protocol canonical keys recur across seeds: the second seed shares 4 two-hop and 1 three-hop records with earlier seeds; the third shares 5 two-hop and 1 three-hop records. These counters include validation and evaluation representatives, so we do not pool seeds as independent new problems. In the third seed's three-hop group, one within-group duplicate has a different reconstructed explicit opposite. This can happen because the original PrOntoQA representation stores no opposite query and its deduplication key therefore cannot distinguish the two reconstructed opposites. The matching query indices are instance-local numbers, not necessarily the same raw query; they do not determine the opposite index. We keep the original deduplication decision and use the first retained representative's opposite for the alternative stopping rule.

Table 15: New-seed partitions. Each seed and hop count starts with 100 generated examples. Deduplication uses the original PrOntoQA representation; validation and evaluation sum to the deduplicated count.

Seed	Hops	Deduplicated	Validation	Evaluation
62,471,894	2	97	11	86
	3	99	12	87
	4	100	9	91
	5	100	16	84
62,471,895	2	97	11	86
	3	100	12	88
	4	100	12	88
	5	100	8	92
62,471,896	2	96	3	93
	3	98	7	91
	4	100	12	88
	5	100	8	92

Table 16: PrOntoQA negative answers at $B = 8$ under a stopping-only intervention. Original stopping requires saturation without q; alternative stopping also accepts a derived $\bar{q}$. The Original and Alternative columns count certified negative answers; the last counts repairable failures. Positive-answer successes are unchanged. The examples and selected policy are fixed, and the ceiling count is computed under the alternative rule.

Hop	NO size	Original	Alternative	Ceiling count	Repairable
2	757	466	757	757	0
3	825	379	825	825	0
4	893	274	893	893	0
5	902	0	405	902	497

Alternative explicit-opposite certification. For PrOntoQA, we reconstruct the explicit opposite $\bar{q}$ from the raw query's predicate, arguments, and polarity. We keep it separate from the original representation, which has no stored opposite query, and verify that the original canonical instance is otherwise unchanged. The alternative retains YES on the query and saturation-based NO, but additionally permits NO as soon as the explicit opposite is derived. The original query-based scoring targets, rule order, and frozen policies remain unchanged. In particular, query relevance still targets q alone; $\bar{q}$ is used only for the new stopping test. Adding $\bar{q}$ to the scoring targets would also change action ordering, preventing isolation of the stopping-rule effect.

All 3,377 original NO examples have derivable opposites; none of the 3,440 YES examples does. Table 16 shows the eight-proposal effect. The five-hop policy leaves 497 failures, all now repairable under the changed protocol. At the original lower positive-control budgets, the same frozen policy under alternative stopping certifies 186/757, 242/825, and 79/893 negative instances for 2-hop/2, 3-hop/4, and 4-hop/4, while their changed-protocol ceilings include every negative instance.

The three new generation seeds exhibit the same protocol dependence: the original selected-policy gap is zero at eight in every hop group; under alternative stopping the only nonzero gaps occur at five hops, with 22/84, 22/92, and 27/92 repairable failures for the three seeds. Because the original deduplication key does not include the reconstructed opposite, two records treated as the same instance under the original protocol can differ under alternative stopping. The evidence retains the reconstructed opposites and overlap diagnostics; we make no claim of independence under both protocols. These controls demonstrate dependence on certification choice without redefining the primary ordering-only counterfactual.

F PROOFWRITER-ADJACENT RULETAKER COMPATIBILITY

RuleTaker (Clark et al., 2020) is used only to check compatibility of the ProofWriter representation, deterministic saturation procedure, local verifier, and open-world stopping rule on an adjacent benchmark. It is not an independent third benchmark family and contains no action-ordering comparison. The compatibility filter follows the supplied annotations: it first checks rule representations for negation-as-failure markers, then checks each remaining question's proof metadata for the closed-world-assumption marker `CWA`, followed by any remaining proof-level negation-as-failure markers. Thus the closed-world category refers to the supplied proof annotation.

In the complete RuleTaker depth-5 test file, 20,192 questions from 967 rulebases are scanned. The strict finite monotone Horn-style subset contains 7,092 questions from 591 rulebases; 8,102 questions are excluded because their rulebases contain negation as failure, and 4,998 because their supplied proofs are marked as using a closed-world assumption. Conversion records no parse or evaluator error. Deterministic saturation matches all 7,092 gold labels, evenly split between 3,546 TRUE and 3,546 FALSE.

A broader but bounded archive scan covers 25 files, at most 100 rulebases per file, and 32,736 questions in total. Of these, 10,392 (31.7%) satisfy the same assumptions, 13,730 (41.9%) are excluded for rulebase-level negation as failure, and 8,614 (26.3%) for proof-level closed-world dependence, assigned in that priority order. These mutually exclusive counts exhaust the bounded scan, but their percentages do not estimate the complete RuleTaker archive.

G SCOPE AND REPRODUCIBILITY BOUNDARIES

The exact criterion applies to finite ground definite Horn-style forward chaining with complete grounded agendas, deterministic transition dynamics, unit proposal cost, strict one-literal extension for accepted actions, and a sound state-based stopping rule. Ranking opportunities depend on these components and the budget. Candidate-recall errors, action generation, pruning, stochastic or learned verifier error, nonmonotonic revision, and changes to the rollout budget are outside this ordering-only intervention. The policy comparisons are proposal-matched but not total-compute-matched because lookahead and beam search perform internal planning.

The empirical no-repair result covers the five retained settings, original rule order, original stopping protocol, and the eight-proposal budget. Threshold membership is known for every primary instance at that budget; additional breadth-first search resolves the eight original numerical cap hits. Table 2 retains uncertainty at one proposal. The expanded ProofWriter scope and rule-order perturbations introduce repairable witnesses; alternative PrOntoQA stopping changes the ceiling itself. Existing original records do not support an evaluation-only table of mean evaluator runtime, mean explored states, branching factor, or memory use, and no such values are inferred here. Because the setting screen used the evaluation pools, all empirical claims and intervals are conditional on the retained settings. PrOntoQA partitioning establishes sample-identifier disjointness, not template- or generated-world-level independence. The RuleTaker check establishes representational compatibility only. Finally, the audit tests whether a successful ordering exists; it does not train a ranker or show that an existing ordering is learnable.

H REPRODUCTION PACKAGE AND VERSION BOUNDARIES

The supplementary archive separates runnable reproduction procedures from archived results. Public benchmark data and the PrOntoQA generator must be obtained separately using the included instructions. Table 17 maps the reported analyses to what the package provides; the instructions give the corresponding commands.

Thus the missing unified experiment entry point is not merely a matter of calling several supplied scripts: some additional studies are represented only by their archived results and applicable evidence checks. Public sample identifiers needed for deterministic partitioning and identifier-seeded policies are preserved; author-specific paths are removed. Packaging and evidence checks do not constitute an independent rerun of every experiment.

Table 17: Reproduction coverage of the anonymous package. Checking a supplied result table or replaying a finite witness does not regenerate the full experiment that produced it.

Reported analysis	Material supplied
Original data preparation and policy comparisons	Runnable adapters, policy-budget comparisons, positive-control resampling, failure-cost analysis, and RuleTaker compatibility checks.
Primary-cohort and full ProofWriter audits	Runnable preparation including explicit negative premises, and an audit of supplied canonical instances under the original stopping rules.
Additional finite witnesses	Grounded instances, successful and failed trajectories, and replay and shortest-distance checks.
New seeds, rule permutations, and alternative stopping	Archived result tables and applicable replayable evidence; no package-contained procedures regenerate these complete studies.
Source alignment, semantic comparisons, and original timing	Archived summaries; no package-contained procedures regenerate all these records.