When Verification Hurts: Asymmetric Effects of Multi-Agent Feedback in Logic Proof Tutoring

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly used for automated tutoring, but their reliability in structured symbolic domains remains unclear. We study step-level feedback for propositional logic proofs, which require precise symbolic reasoning aligned with a learner's current proof state. We introduce a knowledge-graph–grounded benchmark of 516 unique proof states with step-level annotations and difficulty metrics. Unlike prior tutoring evaluations that rely on model self-assessment or binary correctness, our framework enables fine-grained analysis of feedback quality against verified solution paths. We evaluate three multi-agent pipelines with varying solution access: Tutor (partial solution access), Teacher (full derivation access), and Judge (verification of Tutor feedback). Our results reveal a striking asymmetry: verification improves outcomes when upstream feedback is error-prone (<70% accuracy), but *degrades* performance by 4–6pp through over-specification when feedback is already reliable (>85%). Critically, we identify a shared complexity ceiling; no model or pipeline reliably succeeds on proof states exceeding complexity 4–5. These findings challenge the assumption that adding verifiers or richer context universally improves tutoring, motivating adaptive, difficulty-aware architectures that route problems by estimated complexity and upstream reliability.

1 Introduction

Propositional logic is foundational to computer science education, underpinning formal reasoning in programming, algorithms, and digital systems. Yet mastering optimal proof strategies remains challenging for students due to limited access to timely and personalized tutoring (Mayer and Baraniuk, 2025; Inamdar et al., 2025). Intelligent tutoring systems (ITS) achieve learning gains comparable to human tutoring through *step-level feedback*, guiding each inference rather than evaluating only final answers (VanLehn, 2011). However, traditional ITS relies on template-based feedback that poorly adapts to diverse misconceptions (Zerkouk and Chikhaoui, 2025). Large language models (LLMs) offer scalable alternatives, yet their tendency to hallucinate risks incorrect feedback undermining learning (Macina et al., 2023).

Effective step-level tutoring requires more than surface correctness: feedback that is partially correct or misaligned with a student's proof state can reinforce misconceptions and anchor flawed reasoning (VanLehn, 2011). This risk of misaligned feedback is especially acute in propositional logic, where success demands precise symbolic planning, strict rule application, and sustained alignment with the evolving proof state; even small deviations can derail the solution (Tithi et al., 2025). Without external grounding, even capable LLMs risk generating feedback that appears correct but misguides student reasoning, motivating the need for verification mechanisms (Dhuliawala et al., 2024).

Multi-agent verification, where an LLM evaluates another's output, offers a potential safeguard (Zheng et al., 2023). Yet verification without external grounding can degrade performance (Huang et al., 2024). A key factor overlooked in prior work is the degree of solution *information access*: what each agent knows about the correct solution. Solution access may enable precise verification, but it risks revealing answers undermining scaffolding (Daheim et al., 2024). However, how information access shapes verification effectiveness and what failure modes emerge remains unexplored. Moreover, as proof complexity increases, verification errors may compound systematically rather than uniformly, reflecting breakdowns in evaluating intermediate reasoning. We address these gaps through three research questions:

1. *RQ1:* Under what conditions does multi-agent verification improve step-level tutoring accu-

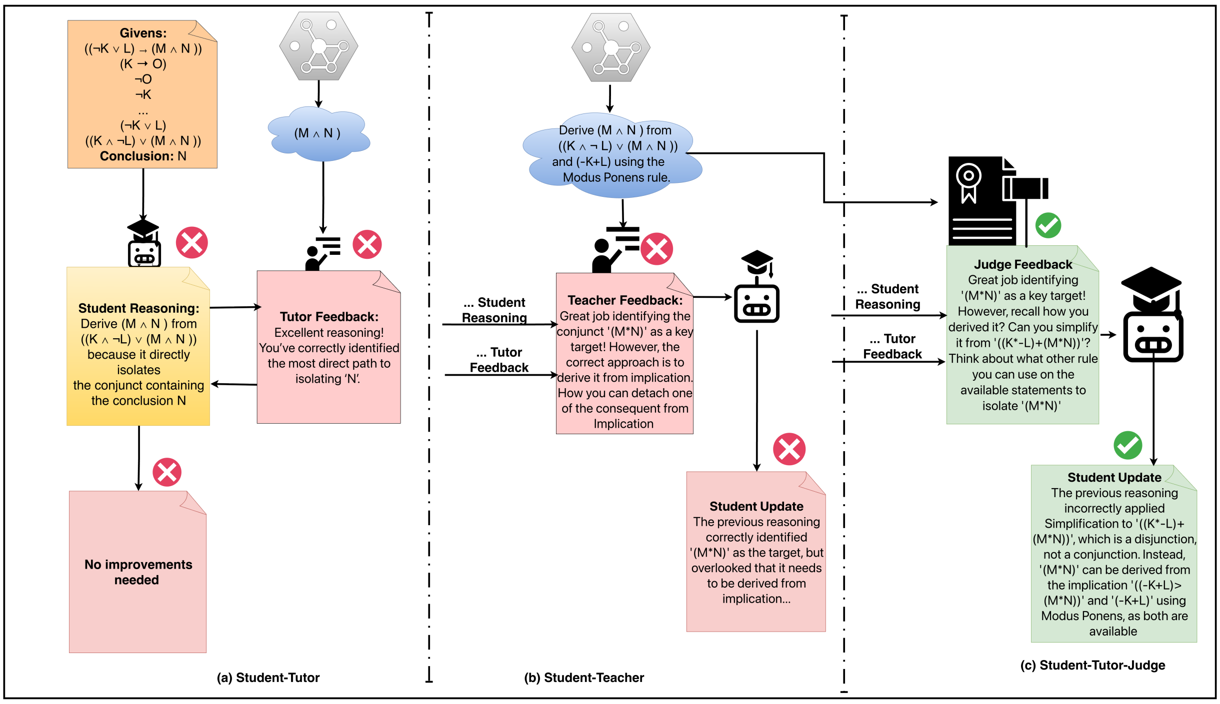

Figure 1: Multi-agent Tutoring with KG-grounded varying hints **(a)** Student solution is partially correct (using incorrect rule) Tutor echoes student's incorrect rule, validates partial correctness sec (6.2.1) **(b)**Teacher reveals solutions prematurely and stduents anchors **(c)** Judge correctly identifies the error in students reasoning (using incorrect rules and statements for derivations)

racy, and how does upstream model quality modulate this effect?

2. *RQ2:* How does varying agents' access to solution information affect feedback quality?
3. *RQ3:* What systematic failure modes emerge across problem complexity (symbolic reasoning difficulty metrics)?

We investigate these questions through three multi-agent tutoring pipelines for propositional logic (Fig 1), where a student agent solves the next step in a proof and receives feedback from tutoring agents. Unlike prior evaluations relying on binary correctness or model self-assessment, we use KG-encoded solution spaces for fine-grained classification of responses as optimal, valid-alternative, or invalid. We operationalize *information access* as derivation context: Tutor receives only the correct step, Teacher receives the full derivation (rule, parents, step), and Judge receives derivation and Tutor feedback. We hypothesize that verification benefit is conditional on upstream feedback quality, helping when error-prone, hurting when already reliable. Our contributions are as follows:

1. **Logic Tutoring Benchmark**: A dataset of 516 unique propositional logic proof states, with a KG-encoded solution space and reasoning difficulty metrics, enabling step-level evaluation beyond binary correctness.
2. **Controlled Multi-Agent Framework**: A tutoring architecture that isolates solution information access from verification by systematically varying agents' access to derivation context.
3. **Conditional Effects of Verification**: Empirical evidence that verification improves error-prone feedback but degrades reliable feedback, with a threshold at ∼70% upstream accuracy and identified failure modes (echo effects, over-specification).
4. **Multi-Dimensional Evaluation**: An evaluation protocol combining KG-encoded correctness and reasoning difficulty analysis, along with human judgment across four pedagogical dimensions.

2 Related Work

2.1 LLM-based scaffolding for tutoring systems

Scaffolding theory (Vygotsky and Cole, 1978; Wood et al., 1976) underlies intelligent tutoring systems (ITS), where step-by-step hints support learning without revealing solutions (Belland, 2017). Traditional ITS achieves strong learning gains comparable to human tutoring (VanLehn, 2011) but remains limited by template-driven feedback that poorly adapts to diverse misconceptions (Zerkouk and Chikhaoui, 2025).

LLMs offer the potential to generate scalable and adaptive feedback when grounded in learning-science principles (Stamper et al., 2024). Prior work shows that LLMs can diagnose student errors and generate pedagogical feedback in college algebra tutors (Reddig et al., 2025), foster active reasoning and self-reflection in advanced computer science courses (Kumar et al., 2024), and support adaptive scaffolding in programming contexts (Scholz et al., 2025). However, two persistent challenges limit their effectiveness in intelligent tutoring systems: (i) LLMs often reveals answers instead of scaffolded guidance (Macina et al., 2023, 2025), and (ii) their feedback frequently hallucinates without external grounding (Liu et al., 2025; Jia et al., 2024). In propositional logic, state-of-the-art models achieve only 86.7% step-level accuracy (Tithi et al., 2025). These limitations highlight the need to understand how external grounding and solution access affect step-level feedback alignment in structured symbolic domains such as propositional logic.

2.2 Multi-Agent Verification

Multi-agent architectures improve LLM reliability through role separation, with generators producing outputs and verifiers assessing their quality.(Zheng et al., 2023; Chudziak and Kostka, 2025). In educational contexts, this separation ensures feedback meets both accuracy and pedagogical standards (Daheim et al., 2024; Phung et al., 2024), addressing over-praise and over-inference errors (dos Santos and Cury, 2025; Chudziak and Kostka, 2025). Prior work shows that role-specialized multi-agent systems improve instructional design (Zhang et al., 2025), and that educational evaluation benefits from incorporating multiple pedagogical roles (Chen et al., 2025). EducationQ further demonstrates that student–teacher–evaluator triads promote effective scaffolding and align with human pedagogical judgment (Shi et al., 2025).

Prior work further shows that self-correction without external grounding can degrade reasoning performance (Huang et al., 2024), and that verification may help weak generators while offering limited or negative gains for strong ones (Zhang et al., 2024). However, these findings are largely based on general reasoning tasks and do not examine step-level tutoring or the role of agents' access to solution information in educational context. This leaves open the question of when verification improves tutoring feedback and when it introduces over-specification or misalignment. We address this gap by introducing a Judge-in-the-loop tutoring architecture that explicitly varies verification and solution access.

2.3 Evaluation Benchmarks

Existing benchmarks for symbolic or logical reasoning primarily evaluate end-to-end proof generation using single-reference solutions, with limited or no annotations for intermediate reasoning steps (Saparov and He, 2022). ProofWriter (Tafjord et al., 2021) assesses only final proof validity, obscuring whether models apply inference rules correctly at each step. FOLIO (Han et al., 2024) provides expert-authored first-order logic problems but penalizes semantically valid alternative derivations, while ProntoQA (Saparov and He, 2022) enables controlled evaluation through synthetic reasoning chains but cannot capture branching decision points where multiple inference rules may apply.

In logic tutoring, effective feedback depends on whether each inference is locally valid and aligned with the learner's current proof state, not merely on final correctness. Research on intelligent tutoring systems, therefore, distinguishes correct, suboptimal, and incorrect actions at the step level (Chi and Wylie, 2014; Corbett, 2001). In contrast, existing symbolic reasoning benchmarks collapse distinct failure modes, incorrect rule use, suboptimal strategies, and state misalignment—into binary outcomes (Saparov and He, 2022; Tafjord et al., 2021), obscuring intermediate reasoning errors.

Evaluating whether verification-augmented tutoring pipelines can correct such errors requires step-level analysis at branching points, a capability absent from current benchmarks. These limitations motivate our study of step-level verification in multi-agent logic tutoring.

3 Task and Dataset

3.1 Task Formulation

A propositional logic proof problem is a tuple $P = (G, C)$, where $G = \{g_1, \ldots, g_n\}$ are premises and C is the target conclusion. A proof $\pi = \langle s_1, \ldots, s_k \rangle$ is a sequence where each s_i is either a premise or is an *intermediate step* derived from preceding premises via inference rule $r \in \mathcal{R}$.

We categorize inference rules into three types: *extraction* (e.g., Modus Ponens, Disjunctive Syllogism), *construction* (e.g., Addition), and *transformation* (e.g., De Morgan's Laws). The complete rule set appears in Appendix A.

Given a partial proof state (G, I, C), premises G, derived intermediates I, and conclusion C, the task is to predict the optimal next step s_{t+1} that minimizes distance to C. A representative proof instance is listed in appendix B.

3.2 Dataset Construction

To instantiate the next-step prediction task with authentic student reasoning, we draw on interaction logs from a propositional logic tutoring system that is used in Undergraduate discrete mathematics course, at a large public U.S. university. Further details and illustrations of the tutor are present in Appendix C. The tutoring system comprises seven levels: a diagnostic assessment level (pre-test), five practice levels with on-demand hints, and a summative assessment level (post-test). We extract data from the five practice levels, where students receive immediate feedback on correctness and may request context-specific hints.

Because propositional logic admits multiple valid proof paths, raw traces contain semantically equivalent states. Following (Rivers and Koedinger, 2017), we apply canonicalization to collapse equivalent states into normalized representations, consistent with prior work on state abstraction in tutoring systems (Barnes and Stamper, 2008; Price et al., 2017).

The final dataset comprises **516 unique proof states** from 32 problems across five difficulty levels from Spring 2023. Each proof-state instance represents a unique student *prestate* including premises, derived intermediates, and target conclusion, ranging from minimal configurations with only three premises to more complex states with 12 intermediates. Instance counts vary across difficulty levels, reflecting natural student progression through the tutoring system. Table 4 provides the complete proof state distribution in the dataset.

3.3 Knowledge Graph Construction

To evaluate next-step predictions against ground truth, we encode each problem's solution space as a knowledge graph encoding all valid inference paths, where nodes represent proof states and directed edges represent valid single-step inferences.

Formally, let $\mathcal{S}$ denote proof states, s_0 the initial state, and s^* the goal state, also known as the conclusion. The knowledge graph is defined as a directed graph $\mathcal{G} = (\mathcal{S}, E)$, between states (S) and edges (E). Edges represent valid inference transitions between current state and subsequent state, s_t and s_{t+1}, respectively such that: $E = \{(s_t, s_{t+1}) \mid s_t \Rightarrow s_{t+1} \text{ via } r \in \mathcal{R}\}$.

Next Step Classification We quantify progress toward the goal using shortest-path distance in the KG, computed via breadth-first search. For any proof state $s \in \mathcal{S}$, its distance to the goal s^* is defined as $d(s) = \min_{\pi: s \rightsquigarrow s^*} |\pi|$, where π ranges over all valid derivation paths. A successor state s_j of s_i is *optimal* if it lies on a shortest path to the conclusion, i.e., $d(s_j) = d(s_i) - 1$.

This definition induces a *three-way classification* of predicted next steps s_{t+1} from the current state s_t, based on shortest-path membership and derivational distance:

1. ***Optimal:*** The next step lies on the shortest path to goal s^* and $(s_t, \hat{s}_{t+1}) \in E$
2. ***Valid-alternative***: There are two cases of predicted next steps that are not optimal:
   (i) *Immediately derivable ($d = 1$):* Next step is directly reachable $(s_t, s_{t+1}) \in E$ but s_{t+1} does not lie on shortest path to s^*.
   (ii) *Non-immediate derivation ($d > 1$):* The next step $\hat{s}_{t+1}$ is not directly reachable from s_t i.e., it can only be derived after more than one inference rule in $\mathcal{G}$.
3. ***Invalid***: The next step does not exist on any inference path to conclusion $(s_t, s_{t+1}) \notin E$

Reasoning Difficulty Metrics We further compute instance-level difficulty metrics that capture the structural difficulty of a predicted next step beyond surface correctness. Specifically, we measure:

1. ***Step Complexity*** measures the intrinsic symbolic difficulty as a weighted function of logical operators, with deeper nesting incurring higher cost.
2. ***Distance to Conclusion*** quantifies the number of remaining inference steps on the shortest path to the goal.
3. ***Derivational Depth*** is computed only for valid alternatives at $d > 1$ to quantify the missing intermediate inferences.

Together with step classification, these difficulty metrics provide a model-independent characterization of reasoning difficulty and form the foundation of our evaluation and analysis. More details are provided in Appendix E.

4 Multi-Agent Tutoring Framework

We compare three feedback pipelines that systematically vary the solution access derived from the KG. All tutoring agents receive the current proof state (premises, intermediates, conclusion) and student response; they differ only in access to KG-derived solution information. Fig 1 and Fig 1 in the appendix explain the difference in hints provided to different tutoring agents.

- **Tutor**: receives only correct next step s_{t+1}
- **Teacher**: receives correct next step s_{t+1} with complete derivation context (rule, parents)
- **Judge**: receives complete derivation along with Tutor feedback

This asymmetric grounding isolates (i) *information access* (Tutor vs. Teacher) and (ii) *verification effects* (Tutor vs. Judge) as independent variables controlling for task difficulty. Figure 1 illustrates the interaction flow.

4.1 Student

The Student agent simulates an undergraduate learner with no solution access following (Shi et al., 2025). Given a proof state, it generates (i) *2–3 candidate steps with rationales*, (ii) *reasoning*, (iii) *next step prediction*, (iv) *inference rule* , and (v) *parent statements*. This multi-candidate approach reflects least-to-most prompting (Zhou et al., 2022) and pedagogical chain-of-thought (Jiang et al., 2024), externalizing reasoning that tutoring agents can reference when scaffolding.

After feedback, the Student agent revises their response only if errors are identified. Otherwise, no change is made. This selective revision ensures measured improvements reflect genuine feedback effects rather than forced changes. Student agent prompts and a sample student response are shown in Figs 6, 7, and 11.

4.2 Tutor

The Tutor models a teaching assistant (TA) who knows the correct answer but does not know the derivation, a common scenario when TAs lack instructor notes, a more constrained setting than the restricted-teacher condition in Shi et al. (2025). Tutor receives only the optimal next step s_{t+1} without derivation context (rule, parent statements), requiring inference of both the applicable rule and reasoning (Kochmar et al., 2022). It classify student responses as optimal, valid-alternative, or invalid, and provides scaffolded feedback without revealing the solution. This tests whether correctness-only grounding suffices for effective tutoring. Tutor prompt and a sample response are shown in Fig 8 and Fig 12.

4.3 Teacher

The Teacher models an instructor with full solution access, the standard assumption in ITS and hint-generation systems (Corbett, 2001; Barnes and Stamper, 2008) representing the upper bound on information availability. It receives the complete derivation (rule, parents, optimal step s_{t+1}), unlike the Tutor, who must infer the derivation from the answer alone. This full access enables precise evaluation, but risks answer revelation (Daheim et al., 2024). Comparing with Tutor, Teacher isolates the effect of solution access on feedback quality. Teacher prompt and sample responses are shown in Fig 9 and Fig 13.

4.4 Judge

The Judge models a quality-assurance reviewer, following verifier-in-the-loop designs in educational AI (Phung et al., 2024; Shi et al., 2025). It receives full derivation context and Tutor feedback for two-stage verification: (i) evaluating student responses against ground truth, and (ii) checking Tutor feedback for correctness and answer leakage. Unlike prior work relying on model self-assessment, our Judge grounds verification in an external knowledge graph. Based on this, it either enhances or overrides the tutor's feedback, addressing over-praise and over-inference failures common in single-agent systems (Gonnermann-Müller et al., 2025; Zhang et al., 2025). Judge prompt and sample responses are shown in Fig 10, Fig 16, and 15.

4.5 Shared Constraints

All feedback agents adhere to scaffolding principles from learning science, guide discovery without revealing answers (Belland, 2017; Van de Pol et al., 2010), employ Socratic questioning over directives (Chen, 2023), reference student candidates for relevance (Serban et al., 2020), and acknowledge correct reasoning before addressing errors (Shi et al., 2025). Feedback is limited to 2-3 sentences. These constraints are enforced identically across agents, isolating information access as the independent variable.

4.6 Interaction Protocol

Each interaction consists of a single feedback round to isolate pipeline effects. To ensure fair comparison, initial Student responses were generated once and reused across all pipelines; Tutor feedback from S-Tu pipeline was reused in S-Tu-J pipeline. All agents communicate via structured JSON format. Implementation details and quality validation constraints across dialogue generation appear in Appendix H.

5 Experimental Setup and Evaluation

5.1 Evaluation Metrics

We evaluate feedback pipelines at the proof-state level, combining KG-grounded correctness with reasoning difficulty metrics.

Task Performance: We report **Pre** (student-only) and **Post** (after feedback) accuracy (N=516), with correctness defined as predicting the optimal next step. **Learning gain** is Δ = Post − Pre in percentage points (pp). **Tutor Rule Accuracy** measures correct rule prediction given minimal hints, serving as a proxy for upstream reliability.

Reasoning Difficulty Analysis: Beyond binary correctness we decompose non-optimal predictions using KG-based step classification (sec 3.3), computing **Complexity Gap** and **Depth Gap** between predicted and expected optimal steps, where positive gaps indicate underestimation of reasoning difficulty.

Verification Effects: To isolate the impact of verification, we compute the **Unique Improvement Count (UIC)**, capturing instances where the Judge corrects errors that persist under Teacher feedback. We further analyze the complexity of these improvement cases to characterize the regimes in which verification intervenes.

5.2 Experimental Setup

We evaluate seven proprietary and open-source LLMs spanning standard and reasoning-augmented architectures, GPT-4.1, GPT-o3, Gemini-1.5-Pro, Mistral-Large, Qwen-3-32B, Llama-3.3-70B, and Deepseek-R1. This selection covers diverse model families and capability levels, including explicit reasoning models (GPT-o3, Gemini-1.5-pro, and Deepseek-R1), to assess whether reasoning capabilities improve logic solving or tutoring performance.

Each model serves as both a student and a feedback agent (Tutor, Teacher, Judge) across all pipelines. We evaluate on 516 proof states spanning five difficulty levels (sec 3), generating 516 × 7 × 3 = 10,836 dialogues. All experiments use temperature 0.0 via official APIs for reproducibility; fewer than 3% of instances required a retry. Model specifications and sample dialogues are presented in Appendices I and G.

6 Results

We present results addressing our three research questions: *RQ1*: whether and when verification improves tutoring, *RQ2:* how information access affects feedback quality, and *RQ3:* what failure modes emerge with increasing problem complexity. We begin with baseline performance, then analyze post-feedback effects, and conclude with cross-cutting patterns in complexity and failure modes.

6.1 Baseline (Student agent) Reasoning Performance

As shown in Table 1, baseline performance varies markedly across models. DeepSeek and Gemini achieve the highest pre-feedback accuracy (56% and 54.5%), followed by Qwen (46%), while GPT variants and Mistral perform more modestly (20–25%) and LLaMA performs weakest (17%). Examining baseline accuracy alongside Tutor rule accuracy reveals two model groups. We refer to Gemini and DeepSeek, which consistently exhibit high performance on both measures, as *strong models*. In contrast, GPT models, Mistral, Qwen, LLaMA show lower performance on both dimensions and are referred to as *weak models* in this paper.

6.1.1 Reasoning Difficulty Analysis

We categorize baseline errors by step complexity and derivational depth, distinguishing immediately derivable valid alternatives (d=1) from requiring more than one inference step (non-local, d>1) in the current proof state (Table 2). Strong models (Gemini, DeepSeek) exhibit balanced error distributions across these categories (e.g., Gemini: 20.23% vs. 18.48%), while weaker models concentrate errors at d>1.

All models underestimate problem difficulty, with positive complexity gaps (1.21–2.34) that widen for non-local derivation d>1: (1.40–2.34) relative to immediately derivable alternatives (d=1: 1.21–1.89). This pattern indicates a preference for simpler, yet non-local reasoning over complex, immediate inferences as models deviate from the op-

Table 1: Pre- and Post-feedback accuracy, learning gain, UIC and complexity across feedback pipelines ($N = 516$)

Model	Pre (%)	Rule Acc.	Post (%)			Δ	Mean Complexity			UIC over Teacher	
	Student	Tutor (%)	Tutor	Teacher	Judge	pp	Tutor	Teacher	Judge	UIC	Gap
GPT-4.1	21.31	58.30	42.82	61.24	**75.38**	54.07	1.8	2.21	2.7	89	1.63
GPT-o3	24.80	60.10	48.25	61.24	**75.58**	50.78	2.04	2.26	2.65	84	1.4
Gemini-1.5-pro	52.52	85.8	73.92	**78.79**	72.95	20.43	3.73	3.17	3.65	23	2.22
Deepseek-R1	56	84.9	76.55	**84.30**	79.84	23.84	3.49	3.06	3.37	17	3.19
Mistral-Large	23.74	54.80	38.41	41.69	**56.37**	32.63	1.86	1.85	2.43	103	0.95
Qwen-3-32B	46	70.20	60.36	63.63	**66.90**	20.9	2.29	2.19	2.6	39	1.59
Llama-3.3-70B	17.98	24.70	24.30	34.38	**39.32**	21.34	2.06	2.06	2.33	55	0.41

pp denotes percentage-point change (Δ = Post − Pre); **UIC** (Unique Improvement Count) counts instances where only the Judge improves over Teacher feedback. **Mean Complexity** reports the average proof-state complexity of improved instances; **Gap** is the difference between predicted and optimal complexity; smaller values indicate better alignment, 0 is ideal

Table 2: Mean baseline complexity and depth gap by outcome

Model	Outcome	Acc.	Complexity		Depth
	Valid Alt.	%	Pre.	Gap	Gap
GPT-4.1	(d=1)	15.16	1.87	1.65	0.00
	(d>1)	49.41	1.70	1.41	1.79
GPT-o3	(d=1)	13.37	2.0	1.55	0.00
	(d>1)	48.64	1.79	1.40	1.82
Gemini	(d=1)	20.23	2.36	1.57	0.00
	(d>1)	18.48	2.45	2.18	1.77
Mistral	(d=1)	14.09	1.97	1.27	0.00
	(d>1)	49.42	1.54	1.83	1.94
Deepseek	(d=1)	20.15	2.01	1.37	0
	(d>1)	20.34	2.65	2.34	1.71
Qwen	(d=1)	11.63	1.56	1.89	0
	(d>1)	41.09	1.80	2.25	1.82
Llama	(d=1)	9.28	1.55	1.21	0
	(d>1)	51.97	1.85	1.53	2.03

d=1 immediately derivable steps; d>1 require missing intermediate inferences. **Pre.** is mean baseline complexity of predicted steps. **Complexity, Depth Gap** measure deviation from the optimal step (lower is better)

timal path. Logistic regression confirms this trend: both complexity and distance are strong negative predictors of correctness (complexity: $\beta \in [-0.30-$ $-0.55]$; distance: $\beta \in [-0.32 - 0.54]$, respectively all $p < .001$), revealing a systematic bias against complex immediate derivations. ***This addresses RQ3 on failure modes under increasing difficulty.*** Full regression statistics are reported in Table 6 in the Appendix.

6.2 Post Feedback Performance

Table 1 shows that verification yields large gains for weak Tutor agents (Rule Acc. <70%: GPT-4.1, GPT-o3, Mistral, Llama), averaging +35pp, but *degrades* performance for strong Tutor agents (Gemini, Deepseek; Rule Acc. >84%) by approximately −5pp. This conditional benefit addresses ***RQ1: Verification helps when upstream feedback is error-prone, but over-specifies when it is already reliable.*** This effect further correlates with complexity: weak Tutor agents improve only on low-complexity problems (mean 1.8–2.1), while the Judge agents extend gains to harder problems (mean 2.4–2.7). Strong Tutor agents already succeed at high complexity (mean 3.5–3.7), leaving limited scope for verification.

6.2.1 Tutor Failure Modes

Manual inspection of Tutor feedback identifies two systematic failure modes underlying the Tutor agent's feedback. The first failure is the ***echo effect*** (Fig 1, 15, and 16), which occurs when the tutor is unable to predict the correct rule and anchors on the student's incorrect rule, generating erroneous feedback. Our manual analysis reveals that echo failure also correlates with Tutor rule accuracy, weak models echo frequently, strong models relatively less.

The second failure, ***partial correctness***, occurs when student agents predicts correct step via flawed reasoning. Tutor agents lacking derivational context validate the faulty logic, reinforcing misconceptions. This pattern appears across all models, demonstrating that LLMs are not capable of distinguishing reasoning failure from outcomes. ***Verification mitigates aforementioned failures through ground-truth access, by overriding echoed errors and reasoning flaws masked by partially correct responses addressing conditional verification benefit in RQ1.*** These findings also emphasize that LLM tutors require a complete derivational context for symbolic reasoning problems to generate reliable feedback directly addressing ***RQ2***.

6.2.2 Teacher Feeback Limitations

Full solution access (Teacher agents) improves feedback on harder instances for some models, but not uniformly across all settings. Among improved cases, Teacher mean complexity (1.85–3.17; Table 1) exceeds Tutor primarily for GPT-based models, while Tutor reaches comparable or higher complexity for others. Logistic regression shows Teacher × Complexity interactions are significant mainly for strong models (Gemini: $\beta = +0.65$, DeepSeek: $\beta = +0.36$, $p < .001$; Table 6), directly addressing ***RQ2 that full access benefits only models able to utilize derivational context.***

Premature Revelation: Manual inspection of Teacher feedback reveals a systematic failure mode: feedback often reveals the correct step (Fig14) or over-specifies hints, leading the student agent to anchor on the disclosed answer without correcting the underlying reasoning (rules and parent statements). This results in surface-level correctness rather than genuine reasoning repair. ***The pattern exposes a core limitation of LLMs in symbolic domains: models tend to follow feedback content rather than integrate it into their reasoning process.*** Consequently, full solution access presents a trade-off; enabling precise guidance while increasing the risk of superficial alignment.

Ablation: *Verifier on Teacher Feedback* To further isolate the role of verification, we applied the same Judge agent on Teacher feedback to mitigate premature solution revelation. This yields no consistent benefit: verifier-on-Teacher agent improves performance only for DeepSeek (+4pp) and is neutral or harmful for all other models. These results confirm that verification effectiveness depends on upstream *error rate*, not *information access*: the Judge agent corrects erroneous guidance but does not enhance already reliable feedback. Moreover, verification cannot undo premature step revelation; once disclosed, verification tends to over-specify feedback, leading to unchanged or degraded outcomes. ***This answers RQ1 on conditional verification and upstream model quality.***

6.2.3 Judge Performance

Verification on Tutor agent yields the largest gains for weak models, sample verification responses are in Fig 16 and 15. Judge agent accuracy exceeds Teacher agent by +14pp (GPT-4.1, GPT-o3) and +15pp (Mistral). The ***Unique Improvement Count (UIC)*** quantifies this: weak models show 84–103 unique verification successes after Teacher failure, vs. 17–23 for strong models (Table 1). The ***Complexity Gap*** reveals that verification helps on systematically harder problems (gap 2.2–3.2 for strong and 0.4–1.6 for weak models), indicating that the verification addresses high-complexity edge cases. Regression analysis confirms this pattern, Judge × Complexity interactions are positive and significant across nearly all models (GPT-4.1: $\beta = +0.22$, Gemini: $\beta = +0.61$, Deepseek: $\beta = +0.56$ for all $p < 0.001$; Table 6). ***This demonstrates that verification* mitigates *the complexity penalty when upstream feedback is error-prone, directly addressing RQ1.***

6.2.4 Universal Complexity Threshold

Across all models and pipelines, performance degrades sharply once proof-state complexity exceeds 4–5 (Table 7 in appendix), ***addressing RQ3 on failure modes across problem complexity***. This ceiling is independent of feedback verification: simple states complexity (<2.5) require minimal intervention, mid-complexity states (2.5–4.0) benefit most from verification, and high-complexity states (>4.5) remain challenging for all approaches.

6.3 Qualitative Analysis

Due to space limitations, complete details of the qualitative analysis are available in Appendix J.

7 Conclusion

We present a controlled study of multi-agent LLM tutoring for propositional logic using a knowledge-graph–grounded benchmark of 516 proof states. Our central finding is that verification benefit is *asymmetric*: it improves outcomes by +10–14pp when upstream feedback is error-prone (<70% accuracy), but degrades performance by 4–6pp when feedback is already reliable (>85%). This asymmetry is governed by the proof-state Complexity. Verification helps in intermediate regimes but cannot overcome a shared reasoning ceiling (complexity >4–5) beyond which no pipeline reliably succeeds. We identify three systematic failure modes: echo effects, partial correctness validation, and premature revelation that verification mitigates only under limited conditions. These findings challenge the assumption that adding verifiers or solution access improves tutoring. Instead, effective LLM tutoring requires adaptive pipelines that estimate upstream reliability and route problems by complexity, rather than defaulting to more agents or richer context.

Limitations

This study has several limitations. First, our evaluation is restricted to propositional logic proof tutoring, and the findings may not directly generalize to domains with weaker symbolic constraints or richer semantic structure. While our primary metrics assess the intrinsic properties of feedback independently of the receiver, LLM students may respond differently from human learners; therefore, human-in-the-loop validation would be necessary to substantiate claims about learning outcomes. Second, while our knowledge-graph-based metrics enable fine-grained analysis of step-level reasoning, they rely on a predefined solution space and do not capture open-ended or creative reasoning strategies. Third, our experiments focus on single-step feedback and do not model long-horizon student–tutor interactions or learning gains over time. Moreover, our qualitative analysis is limited to a small expert-reviewed sample (10 responses per model), and broader manual analysis could reveal additional failure modes. Finally, although we analyze multiple large language models, results may vary with future architectures or training regimes.

Ethics Statement

Student interaction data was collected under IRB approval with informed consent. All data was anonymized before analysis. Our LLM-based Evaluation does not involve human subjects beyond annotators, who serve as researchers for the tutoring system used for data generation in this study, as well as co-authors of this study.

Reproducibility

Code and data will be released upon publication. All experiments use temperature 0.0 to ensure deterministic, reproducible results, prioritizing controlled comparison over response diversity. While real-world tutoring systems might benefit from non-zero temperature, our evaluation focuses on measuring the systematic effects of multi-agent architectures under controlled conditions. Model specifications and prompts appear in Appendices I and F.

AI Usage Disclosure

This work studies and evaluates large language models as research objects. We utilized large language models as assistive tools during manuscript preparation, including formatting guidelines and brainstorming organization, mainly, and paraphrasing on a need basis only. We did not use any AI tools for designing, implementing, or executing this research study. All the claims, analyses, hypotheses, and conclusions are developed, verified, and reviewed by the authors. Moreover, no AI tool was used for generating or labeling data, making judgments about data, or making any scientific claims. All implementations were reviewed, validated, and finalized by the authors. The authors take full responsibility for the correctness, originality, and integrity of the work.

References

Tiffany Barnes and John Stamper. 2008. Toward automatic hint generation for logic proof tutoring using historical student data. In *International conference on intelligent tutoring systems*, pages 373–382. Springer.

Brian R Belland. 2017. Instructional scaffolding in stem education: Strategies and efficacy evidence. *Springer Nature*. Comprehensive review of scaffolding strategies in STEM.

Jiaju Chen, Yuxuan Lu, Xiaojie Wang, Huimin Zeng, Jing Huang, Jiri Gesi, Ying Xu, Bingsheng Yao, and Dakuo Wang. 2025. Multi-agent-as-judge: Aligning llm-agent-based automated evaluation with multi-dimensional human evaluation. *arXiv preprint arXiv:2507.21028*.

others Chen. 2023. Artificial intelligence in intelligent tutoring systems toward sustainable education: A systematic review. *Smart Learning Environments*, 10:41. ITS monitor progress, provide feedback, hints, and scaffolding.

Michelene TH Chi and Ruth Wylie. 2014. The icap framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243.

Jarosław A Chudziak and Adam Kostka. 2025. Ai-powered math tutoring: Platform for personalized and adaptive education. In *International Conference on Artificial Intelligence in Education*, pages 462–469. Springer.

Albert Corbett. 2001. Cognitive computer tutors: Solving the two-sigma problem. In *International Conference on User Modeling*, pages 137–147. Springer.

Nico Daheim, Jakub Macina, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2024. Stepwise verification and remediation of student reasoning errors with large language model tutors. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8386–8411.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578.

O Lube dos Santos and D Cury. 2025. Intelligent multi-agent systems with generative ai in education: A systematic literature mapping. *ICERI2025 Proceedings*, pages 9037–9046.

Jana Gonnermann-Müller, Jennifer Haase, Konstantin Fackeldey, and Sebastian Pokutta. 2025. Facet: Teacher-centred llm-based multi-agent systems-towards personalized educational worksheets. *arXiv preprint arXiv:2508.11401*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2024. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.

Jie Huang and 1 others. 2024. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*.

Amogh Inamdar, Uzay Macar, Michel Vazirani, Michael Tarnow, Zarina Mustapha, Natalia Dittren, Sam Sadeh, Nakul Verma, and Ansaf Salleb-Aouissi. 2025. Logiclearner: A tool for the guided practice of propositional logic proofs. *arXiv preprint arXiv:2503.19280*.

Qinjin Jia, Jialin Cui, Ruijie Xi, Chengyuan Liu, Parvez Rashid, Ruochi Li, and Edward Gehringer. 2024. On assessing the faithfulness of llm-generated feedback on student assignments. In *Proceedings of the 17th International Conference on Educational Data Mining*, pages 491–499.

Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li, and Dongsheng Li. 2024. Llms can find mathematical reasoning mistakes by pedagogical chain-of-thought. *arXiv preprint arXiv:2405.06705*.

Ekaterina Kochmar, Dung Do Vu, Robert Belfer, Varun Gupta, Iulian Vlad Serban, and Joelle Pineau. 2022. Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, 32(2):323–349.

Harsh Kumar, Ruiwei Xiao, Benjamin Lawson, Ilya Musabirov, Jiakai Shi, Xinyuan Wang, Huayin Luo, J. Williams, Anna N. Rafferty, John Stamper, and Michael Liut. 2024. Supporting self-reflection at scale with large language models: Insights from randomized field experiments in classrooms. *Proceedings of the Eleventh ACM Conference on Learning @ Scale*.

Rongxin Liu, Julianna Zhao, Benjamin Xu, Christopher Perez, Yuliia Zhukovets, and David J Malan. 2025. Improving ai in cs50: Leveraging human feedback for better learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 715–721.

Jakub Macina, Nico Daheim, Sankalan Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2023. Mathdial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5602–5621.

Jakub Macina, Nico Daheim, Ido Hakimi, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2025. Mathtutorbench: A benchmark for measuring open-ended pedagogical capabilities of llm tutors. *arXiv preprint arXiv:2502.18940*.

Paul Mayer and Rich Baraniuk. 2025. The importance of teaching logic to computer scientists and electrical engineers. *ACM Transactions on Computing Education*, 25(2):1–12.

Tung Phung, Victor-Alexandru Pădurean, Anjali Singh, Christopher Brooks, José Cambronero, Sumit Gulwani, Adish Singla, and Gustavo Soares. 2024. Automating human tutor-style programming feedback: Leveraging gpt-4 tutor model for hint generation and gpt-3.5 student model for hint validation. In *Proceedings of the 14th learning analytics and knowledge conference*, pages 12–23.

Thomas W Price, Yihuan Dong, and Dragan Lipovac. 2017. isnap: towards intelligent tutoring in novice programming environments. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on computer science education*, pages 483–488.

Jennifer M Reddig, Arav Arora, and Christopher J MacLellan. 2025. Generating in-context, personalized feedback for intelligent tutors with large language models. *International Journal of Artificial Intelligence in Education*, pages 1–42.

Kelly Rivers and Kenneth R Koedinger. 2017. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education*, 27(1):37–64.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Niklas Scholz, Manh Hung Nguyen, Adish Singla, and Tomohiro Nagashima. 2025. Partnering with ai: A pedagogical feedback system for llm integration into programming education. In *European Conference on Technology Enhanced Learning*, pages 243–248. Springer.

Iulian Vlad Serban and 1 others. 2020. Automated personalized feedback improves learning gains in an intelligent tutoring system. In *International Conference on Artificial Intelligence in Education*, pages 140–146. Springer. Personalized hints improve learning gains; feedback from ZPD.

Yao Shi, Rongkeng Liang, and Yong Xu. 2025. EducationQ: Evaluating LLMs' teaching capabilities through multi-agent dialogue framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32799–32828, Vienna, Austria. Association for Computational Linguistics.

John Stamper, Ruiwei Xiao, and Xinying Hou. 2024. Enhancing llm-based feedback: Insights from intelligent tutoring systems and the learning sciences. In *International Conference on Artificial Intelligence in Education*, pages 32–43. Springer.

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

Sutapa Dey Tithi and 1 others. 2025. The promise and limits of llms in constructing proofs and hints for logic problems in intelligent tutoring systems. *Computers & Education: Artificial Intelligence*, 8:100304. Evaluates LLM accuracy on stepwise proof construction and optimal next-step hints.

Janneke Van de Pol, Monique Volman, and Jos Beishuizen. 2010. Scaffolding in teacher–student interaction: A decade of research. *Educational Psychology Review*, 22(3):271–296. Comprehensive review of scaffolding in teacher-student interactions.

Kurt VanLehn. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221. Step-based ITS achieve effect sizes of 0.75-0.80, comparable to human tutoring (0.79).

Lev Semenovich Vygotsky and Michael Cole. 1978. *Mind in society: Development of higher psychological processes*. Harvard university press.

David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100. Foundational work on scaffolding and just-in-time interventions.

Mohamed Zerkouk and Belkacem Chikhaoui. 2025. A comprehensive review of ai-based intelligent tutoring systems: Applications and challenges. *arXiv preprint arXiv:2507.18882*. ITS can improve student performance; CLT informs stepwise hint design.

Xueqiao Zhang, Chao Zhang, Jianwen Sun, Jun Xiao, Yi Yang, and Yawei Luo. 2025. Eduplanner: Llm-based multi-agent systems for customized and intelligent instructional design. *IEEE Transactions on Learning Technologies*.

Yifan Zhang and 1 others. 2024. Small language models need strong verifiers to self-correct reasoning. In *Findings of the Association for Computational Linguistics (ACL)*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Inference Rule List

We employ a fixed set of propositional inference rules used in Logic Tutor for the dataset. The short names were used for consistent response generation and evaluation. The complete list of inference rules, along with short names and derivations are provided below in Table 3.

Table 3: Propositional inference rules used in this work.

Abbrev.	**Rule Name**	**Form**
MP	Modus Ponens	$P \to Q,\ P \Rightarrow Q$
MT	Modus Tollens	$P \to Q,\ \neg Q \Rightarrow \neg P$
Conj	Conjunction	$P,\ Q \Rightarrow P \land Q$
Simp	Simplification	$P \land Q \Rightarrow P$ (or Q)
Add	Addition	$P \Rightarrow P \lor Q$
DS	Disjunctive Syllogism	$P \lor Q,\ \neg P \Rightarrow Q$
HS	Hypothetical Syllogism	$P \to Q,\ Q \to R$ $\Rightarrow P \to R$
Impl	Implication	$P \to Q \equiv \neg P \lor Q$
DN	Double Negation	$P \equiv \neg\neg P$
CP	Contraposition	$P \to Q \equiv \neg Q \to \neg P$
Com	Commutation	$P \lor Q \equiv Q \lor P$
Assoc	Associativity	$(P \lor Q) \lor R$ $\equiv P \lor (Q \lor R)$
Dist	Distribution	$P \land (Q \lor R)$ $\equiv (P \land Q) \lor (P \land R)$
CD	Constructive Dilemma	$(P \to Q), (R \to S), P \lor R$ $\Rightarrow Q \lor S$
Equiv	Equivalence	$P \leftrightarrow Q$ $\equiv (P \to Q) \land (Q \to P)$

B Representative Proof State

This appendix presents an illustrative propositional logic proof instance Fig 2 annotated with intermediate derivations and agent-specific hints. Intermediate steps are derived by applying valid inference rules to previously established statements, with each step indexed and linked to its parent statements and rule application. This explicit structure

Representative proof Instance

Givens

- (1) $((S \to Y) \lor (I * Q))$
- (2) $((I \land Q) \to D)$
- (3) $\neg D$
- (4) $((S \to Y) \to D)$

Intermediate Steps

- (5) $\neg(S \to Y)$ [MT: (3), (4)]
- (6) $\neg(I \land Q)$ [MT: (3), (2)]
- (7) $(S \to Y)$ [DS: (6), (1)]

Conclusion: Y

Tutor Hint

$\neg(\neg S \lor Y)$

Teacher & Judge Hint

Derive $\neg(\neg S \lor Y)$ from $\neg(S \to Y)$ using the Implication rule.

Figure 2: Illustrative proof instance showing intermediate derivations and differential hint access for Tutor and Teacher agents.

exposes the shortest derivation path toward the conclusion while preserving alternative valid reasoning trajectories.

The Tutor is provided only with the KG-verified optimal next symbolic step, enabling guidance that focuses on local progression without access to derivational context. In contrast, the Teacher additionally receives the rule and parent statements used to derive this step, allowing feedback to reference structural reasoning without revealing the answer directly. This distinction isolates the effect of solution access: the Tutor operates under minimal grounding, while the Teacher leverages full derivational context to support more informed intervention. The representative proof state is available in Fig 2.

C Illustrative Logic Tutor Proof Interaction

Figure 3–5 illustrates a representative student interaction in the propositional logic tutor. These screenshots demonstrate forward chaining, rule application, and goal completion within the tutor interface.

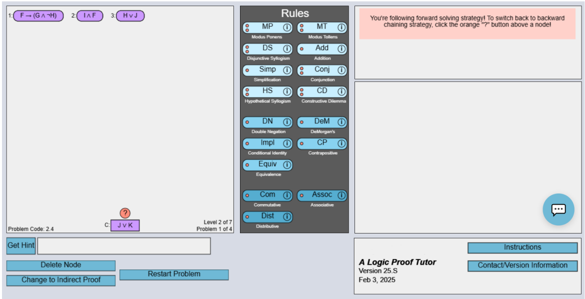

Figure 3: **Initial proof state and goal specification.** The student is presented with the premises (top left) and the target conclusion ($J \lor K$) at the bottom. Available inference rules are displayed on the middle. At this stage, no intermediate steps have been derived, and the student must choose a productive forward step toward the goal.

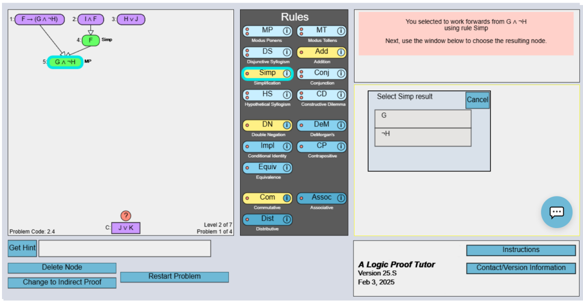

Figure 4: **Rule application with guided simplification.** After deriving an intermediate conjunction via Modus Ponens, the student applies the *Simplification* rule. The interface prompts the learner to select the appropriate resulting literal (G or $\neg H$), illustrating fine-grained, step-level decision making supported by rule constraints.

D Dataset Distribution

Table 4: Dataset distribution across practice levels (2-6)

Difficulty	Proof States	Avg. Statements
2 (Introductory)	48	5.86
3 (Basic)	111	7.73
4 (Intermediate)	148	7.94
5 (Advanced)	85	5.64
6 (Expert)	95	6.41
Total	**516**	**6.72**

*Levels 1 and 7 excluded because of unavailability of hints.

E Knowledge Graph based Evaluation Metrics

This appendix provides formal definitions and illustrative examples of the graph-based metrics used to evaluate next-step reasoning quality. All metrics are computed with respect to the knowledge graph (KG) constructed for each problem.

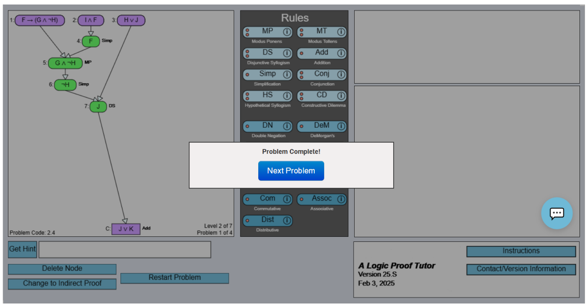

Figure 5: **Successful proof completion.** The student derives J via Disjunctive Syllogism and applies the *Addition* rule to reach the target conclusion $J \vee K$. The system confirms completion, reinforcing correct rule sequencing and alignment with the goal state.

E.1 Step Complexity

Beyond derivational validity, *Step Complexity* quantifies the intrinsic symbolic difficulty of a single inference step based on the structural complexity of the derived statement. We define step complexity as a function

$$c : E \to \mathbb{R}^+,$$

computed directly from the syntactic structure of ϕ. Let $\mathcal{O}(\phi)$ denote the set of logical operators in ϕ, and let $\text{depth}(o)$ denote the nesting depth at which operator o occurs. The complexity of the step is defined as:

$$c(s_i, s_j) = \sum_{o \in \mathcal{O}(\phi)} w(o) \cdot \alpha^{\text{depth}(o)},$$

where $w(o)$ is an operator-specific base weight (e.g., $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$) and $\alpha > 1$ controls the penalty for nested structure. For example, $c(F) < c(A + B) < c(A \rightarrow (B + C)$, because no operator is involved, only 1 operator is involved, and 2 operators involved along with nesting.

E.2 Distance to Conclusion

For any proof state $s \in \mathcal{S}$, its distance is defined as the length of the shortest derivation path to a terminal state s^*: $d(s) = \min_{\pi : s \rightsquigarrow s^*} |\pi|$ computed via breadth-first search over the KG. A predicted next step is considered *optimal* if it strictly reduces the distance to the goal by one.

Example If $d(s_t) = 4$ and a predicted successor state $\hat{s}_{t+1}$ satisfies $d(\hat{s}_{t+1}) = 3$, the step is distance-optimal. A valid step leading to a state with $d(\hat{s}_{t+1}) = 5$ is considered a valid it is non-optimal.

E.3 Derivational Depth

Derivational depth measures how many missing intermediate inference steps separate a predicted statement from the current proof state. This metric distinguishes immediately derivable steps from those that implicitly skip required reasoning and thus cannot be derived in one sentence. Formally, if for a predicted statement ϕ not present in the immediate successor states of s_t, we define derivational depth as the minimum number of edges required to derive ϕ from s_t in the KG.

Example If ϕ can be derived from s_t only after two intermediate inferences, its derivational depth is 2. Predictions with depth 1 correspond to directly derivable steps, while higher depths indicate premature or unsupported jumps in reasoning.

E.4 Parent Statements and Rule Justification

Each edge $(s_i, s_j) \in E$ in the KG is annotated with the inference rule applied and the minimal set of parent statements used in the derivation. This enables explicit verification of whether a model-generated step is justified by the available premises.

Example A model predicting statement C with rule Modus Ponens must correctly identify both $(A \rightarrow C)$ and A as parent statements. Predictions that produce a valid statement but fail to align with the correct parents or rule are flagged as unjustified.

F Prompt Design

This section provides the complete prompts used for each agent in our multi-agent tutoring system. All prompts use structured JSON response formats to ensure consistent parsing.

Our prompt design translates established pedagogical principles into operational constraints for LLM agents. Each agent's prompt encodes specific instructional strategies grounded in learning science research, ensuring that generated feedback aligns with effective tutoring practices. We describe the key design decisions below.

F.1 Student Prompt

The base system message includes the Role and Task information along with instructions, whereas the user message includes the problem instance itself. The prompt template is provided below in Fig 6.

Student Prompt

Role: You are a Student in an undergraduate Discrete Structures course solving a propositional logic proof. Your task is to produce the *single most optimal next step* that advances the proof toward the conclusion.

Task:
1. Review the givens and intermediate steps.
2. Propose 2–3 candidate next steps.
3. Select the candidate that most directly advances toward the conclusion.
4. Justify your choice and output the selected next step.

Constraints:
- Output exactly one next step in *symbolic notation only*.
- Use only predefined inference rules (e.g., MP, MT, Conj, DS).
- Parent statements must be actual expressions, not line numbers.

Response Format:
- `CANDIDATES`: 2–3 candidate steps with brief justification
- `REASONING`: Why the selected step is optimal
- `NEXT_STEP`: Symbolic expression
- `RULE`: Inference rule (short name)
- `PARENT_STATEMENTS`: Supporting expressions

Figure 6: Base System Prompt for Student

F.2 Student Update Prompt

After receiving feedback, the Student Update agent simulates a learner revising their approach. The prompt requires the student to: (1) identify what aspect of the feedback addresses their error, (2) generate revised candidates incorporating the guidance, and (3) produce a corrected next step.

Critically, if the original response was marked "Correct," the agent responds with "No Improvements Needed" rather than artificially generating changes. This prevents the evaluation artifact of forced revisions when none are warranted, ensuring that measured improvement rates reflect genuine learning effects. The base system prompt is provided in Fig 7.

Student Update Prompt

Role: You are a Student revising your previous next-step attempt after receiving feedback on a propositional logic proof.

Task:
1. Review the feedback to determine whether errors are present.
2. If the prior response is correct, return `No Improvement Needed`.
3. Otherwise, revise your reasoning and propose exactly one improved next step.

Constraints:
- Revise only the *immediate next step*; do not extend the proof.
- Output symbolic expressions only; use predefined inference rules.
- Do not infer or reveal the full solution.

Response Format:
- `REVISED_REASONING`: Brief reflection or `No Improvement Needed`
- `IMPROVED_STEP`: Symbolic expression or `No Improvement Needed`
- `BETTER_RULE`: Inference rule (short name)
- `PARENT_STATEMENTS`: Supporting expressions or `No Improvement Needed`

Figure 7: Base system prompt for student update

F.3 Tutor Prompt

The Tutor operates under *minimal hint access*: it receives only the correct next step without knowledge of the complete solution path. The prompt enforces a critical **planning-before-feedback** requirement: the Tutor must first generate an internal derivation plan explaining how the correct step is derived (identifying the rule and parent statements) before producing student-facing feedback. This design decision ensures the Tutor develops a genuine understanding of the solution rather than pattern-matching, aligning with research showing that scaffolded feedback improves instructional quality (Serban et al., 2020). The prompt is illustrated in Fig 8.

Tutor Prompt

Role: You are a Tutor evaluating a student's proposed next step in a propositional logic proof, with access to the KG-derived optimal step.
Task:
1. Analyze how the optimal step is derived (rule and parent statements).
2. Evaluate the student's candidates, reasoning, and chosen next step.
3. Classify the student's step as *Correct*, *Suboptimal*, or *Incorrect*.
4. Provide brief, scaffolded feedback guiding the student toward the optimal step.

Constraints:
- Do not reveal the optimal step, its rule, or parent statements.
- Acknowledge what the student did correctly before addressing errors.
- Use Socratic questions to guide reasoning; keep feedback concise (2–3 sentences).
- Use predefined inference rule short names only.

Response Format:
- `STUDENT_ERRORS`: Brief explanation or `Correct`
- `NEXT_STEP_CORRECTNESS`: `Correct` / `Suboptimal` / `Incorrect`
- `TUTOR_FEEDBACK`: Scaffolded guidance without answer revelation

Figure 8: Base System prompt for Tutor

F.4 Teacher Prompt

The Teacher agent has access to the complete next step and provides scaffolded feedback without revealing the answer. The prompt is illustrated in Fig 9.

Teacher Prompt

Role: You are a Teacher evaluating a student's proposed next step in a propositional logic proof, with access to the complete solution (`KNOWLEDGE_BASE_STEPS`).

Task:
1. Compare the student's response against the knowledge-base solution.
2. Identify errors in the student's logic, rule usage, or reasoning.
3. Classify the student's next step as *Correct*, *Suboptimal*, or *Incorrect*.
4. Provide brief, scaffolded feedback guiding the student toward the correct solution.

Constraints:
- Do not reveal the exact next step, rule, or parent statements from the solution.
- Acknowledge correct aspects of the student's attempt before addressing errors.
- Use Socratic questions to guide reasoning; keep feedback concise (2–3 sentences).
- Refer to the student's candidates when relevant.
- Use predefined inference rule short names only.

Response Format:
- `STUDENT_ERRORS`: Brief explanation or `Correct`
- `NEXT_STEP_CORRECTNESS`: `Correct` / `Suboptimal` / `Incorrect`
- `TEACHER_FEEDBACK`: Scaffolded guidance without answer revelation

Figure 9: Base System prompt for Teacher

F.5 Judge Prompt

The Judge agent is tasked with thoroughly evaluating both the student's and the teacher's responses to the logic proof problem. While the Teacher has access to the correct step and is therefore not asked to verify its own guidance, the Judge is specifically instructed to examine and identify any errors or inaccuracies present in both the student's and the teacher's responses.

Equipped with comprehensive access to the correct derivation steps from the knowledge base, the Judge carefully analyzes the reasoning behind both responses. Its role is to provide targeted evaluations and, where necessary, concise and actionable feedback to ensure alignment with the correct logic step. This approach ensures a robust layer of verification and strengthens the overall quality and pedagogical value of the feedback offered to the student. The prompt is illustrated in Fig 10.

G Illustrative Agent Responses

This appendix presents representative responses generated by the Student, Tutor, Teacher, and Judge agents to concretely illustrate how each role operationalizes its assigned information access and reasoning constraints. These examples are intended to complement the quantitative analysis by providing qualitative insight into agent behavior across the tutoring pipeline. The responses are color-coded, red represents an incorrect student response or tutoring agent feedback, while blue represents correct responses and feedbacks.

G.1 Student Agent Response

We show a representative Student agent response to demonstrate candidate generation, explicit reasoning, and next-step generation. The example corresponds to a single proof instance and highlights how intermediate reasoning is externalized for downstream tutoring and verification. The sample response is shown in Fig 11.

Judge (Verifier) Prompt

Role: You are an expert pedagogical AI Judge for propositional logic proof problems, with access to the complete solution (`KNOWLEDGE_BASE_STEPS`). You evaluate both the student's proposed next step and the Teacher's feedback. **Task:**

1. Compare the student's response against the knowledge-base solution.
2. Identify errors in the student's reasoning, if any.
3. Classify the student's next step as *Correct*, *Suboptimal*, or *Incorrect*.
4. Evaluate whether the Teacher's feedback correctly guides the student.
5. Either enhance the Teacher's feedback or override it with corrected guidance.

Constraints:

- Do not reveal the exact next step, rule, or parent statements from the solution.
- Acknowledge correct aspects of the student's attempt before addressing errors.
- Use Socratic questions to guide reasoning; scaffold rather than instruct.
- Override Teacher feedback if it is incorrect, misleading, or reveals the solution.
- Keep final feedback concise (2–3 sentences) and encouraging.
- Use predefined inference rule short names only.

Response Format:

- `STUDENT_ERRORS`: Brief explanation or `Correct`
- `NEXT_STEP_CORRECTNESS`: `Correct` / `Suboptimal` / `Incorrect`
- `TEACHER_FEEDBACK_CORRECTNESS`: Assessment of Teacher feedback
- `JUDGE_ACTION`: `Enhanced` or `Overridden`
- `FINAL_FEEDBACK`: Judge-approved scaffolded guidance

Figure 10: Base System prompt for Judge

Sample Instance

Givens:

- $((\neg K \lor L) \to (M \land N))$
- $(K \to O)$
- $\neg O$

Intermediates:

1. $\neg K$ (MT: 2,3)
2. $(\neg K \lor L)$ (Add: 4)
3. $(M \land N)$ (MP: 1,5)

Conclusion: N

Corresponding student Response

Candidate Steps:

- M: potentially useful for deriving N from $(M \land N)$.
- $\neg M$: less direct without a clear derivation.
- $(M \land N)$: already derived; insufficient alone to reach N.

Reasoning: Applying a rule directly to $(M \land N)$ is the most efficient path to the conclusion. Since $(M \land N)$ is available, simplification yields the target.

Next Step: N

Rule: Simp

Parent Statement: $(M \land N)$

Figure 11: Sample problem instance and corresponding **correct** student response

G.2 Tutor Agent Response

This example illustrates a Tutor agent interaction under *partial solution access*. The Tutor receives only the next step and must provide scaffolded feedback without access to the full derivation context or rule justification. The sample response is shown in Fig 12.

Tutor Response Interaction

Problem Instance (Excerpt)

- $(Z \to (\neg Y \to X))$
- $(Z \land \neg W)$
- ...

Derived Intermediates (Excerpt)

- $\neg W$ (Simplification)
- Z (Simplification)
- $(\neg Y \to X)$ (Modus Ponens)
- ...

Student Response

- **Candidates:**
  - $(Y \lor X)$: *conjunction aiming to reach* $(X \lor Y)$...
  - ...
- **Reasoning:** $(X \lor Y)$ is optimal because it could be obtained via Addition from either X.
- **Predicted Step:** $(X \lor Y)$
- **Rule:** Add
- ...

Tutor Hint

- **Hint:** $(Y \lor X)$

Tutor Feedback

- **Tutor Rule:** Addition
- **Student Assessment:** incorrect
- **Feedback:** *"You have correctly identified the next step..."*

Figure 12: Tutor feedback (partial hint) on **correct** student response

G.3 Teacher Agent Response

This example illustrates a Teacher agent with full derivational access, including the correct rule and parent statements. Unlike the Tutor, the Teacher can precisely diagnose suboptimal reasoning and scaffold guidance using explicit structural knowledge while avoiding direct answer revelation. The sample response is shown in Fig 13.

Teacher Response Interaction

Problem Context (Excerpt)

- $(F \to (G \land \neg H)$
- $(I \land F)$
- $(H \lor J)$
- ...

Student Response

- **Candidates**

– (J+K) by Addition from J
– ...

- **Reasoning:** (J+K) is optimal because it directly derives ...
- ...

Teacher Hint

Derive (J+K) from J using the Addition rule.

Teacher Response

- **Next-Step Correctness:** Correct
- **Student Errors:** None
- **Teacher Feedback:** *You correctly identified that Addition on J directly achieves the conclusion...*

Figure 13: Teacher feedback (complete hint) on **correct** student response

Teacher Reveals Solution

Givens:

- $(S > D) \lor I$
- $(-S \lor Q) > Y$
- $\neg D$
- $\neg D > \neg I$

Student Response

- **Reasoning:** MT on $\neg D > \neg I$ and $\neg D$ is needed to get $\neg I$ to be used later with $(S > D) \lor I$ for DS
- **Next Step:** $\neg I$
- **Rule Used:** MT

Knowledge-Graph Correct Hint:

Derive $\neg S$ from $(S > D)$ and $\neg D$ using *Modus Tollens (MT)*.

Teacher Response

Teacher Feedback: Great job identifying the relevant candidates! You correctly spotted that $(\neg D \rightarrow \neg I)$ and $\neg D$ could be useful, but there's a small mix-up with the rule. Recall: MT requires a negated *consequent*, while MP works with a *true antecedent*. How you might use MP on $(S > D)$ and $\neg D$ to get $\neg D$.

Figure 14: Teacher (complete hint) feedback reveals solution on **incorrect** student response

G.4 Judge Verification Response

This example illustrates a complete tutoring interaction, showing how the student's reasoning is evaluated by a Tutor with partial solution access, refined by a Judge with full derivational access. The instance highlights how verification can override pedagogically suboptimal feedback when the tutor does not produce good feedback. The sample response is shown in Fig 15 and Fig 16.

Judge Verification Improves Tutor Echo

Problem Context (Excerpt)

- $(S \rightarrow D)$
- $((\neg S \lor Q) \rightarrow Y)$
- $\neg D$
- $(\neg D \rightarrow \neg I)$
- ...

Student Response

- **Candidates:**
  – Simplify $S \rightarrow D$ to get $\neg S$
  – ...
- **Reasoning:** *$\neg S$ is optimal because we already have $(S \rightarrow D)$ from INTERMEDIATE STEPS, and simplifying it to $\neg S$ allows us to construct $(\neg S \lor Q)$...*
- **Rule:** Simp

Tutor Hint

$\neg S$

Tutor Response

- **Tutor Rule:** Simp - *echo failure*
- **Student Assessment:** Correct
- **Tutor Feedback:** *Excellent reasoning! You correctly identified how simplifying $(S \rightarrow D)$ helps set up the next step toward the conclusion...*

Judge Verification

- **Tutor Feedback Correctness:** Incorrect
- **Judge Action:** Overridden Teacher Feedback
- **Final Judge Feedback:** *Great job recognizing that you need -S to proceed! However, be*

careful with the rules: you can't directly use Simp on a conditional like $(S \rightarrow D)$, think about what rule you can use on $(S \rightarrow D)$ to turn it into a form more suitable for Simp...

Figure 15: Juge Verification Response, correcting Incorrect Tutor feedback (anchoring on incorrect student reasoning), Judge verification overriding incorrect feedback.

Judge verification helps Tutor Ambiguous Feedback

Relevant Givens (subset):

$((\neg Q+P) \rightarrow S)$, $(\neg T)$, $((\neg T*S) \rightarrow \neg R)$

Student Response

- **Reasoning:** DS on $((\neg Q + P) \rightarrow S)$ and $(\neg Q + P)$ is optimal because it directly yields S, which can combine with $\neg T$ to form $(\neg T * S)$ and move toward deriving $\neg R$.
- **Proposed Step:** S
- **Applied Rule:** Disjunctive Syllogism (DS)

Tutor Hint

S

Teacher Feedback

- **Tutor Rule:** Disjunctive Syllogism (DS) (echo failure)
- **Feedback Text:** *Excellent reasoning! You've identified the most direct path to derive a key component. Now, how might this newly derived statement interact with the other givens to progress toward the conclusion?* (ambiguous and generic feedback)

Judge (Verifier) Response

- **Teacher Feedback Correctness:** *Vague*
- **Judge Action:** *Overridden teacher's feedback*
- **Judge Feedback:** *Great job spotting that S is a key component! You're right that deriving S brings you closer to using $((\neg T * S) \rightarrow \neg R)$. Now, consider: which rule actually applies to $((\neg Q + P) \rightarrow S)$ and $(\neg Q + P)$ to derive S? How might this step help you connect to the other givens, like $\neg T$?*

Figure 16: Judge Verification Response, Tutor echoes incorrect student rule and generates ambiguous and vague feedback - Judge with full solution access helps

H Implementation Details

Response Constraints. We standardize inference rule references using abbreviations (MP, MT, DS, HS, DeM, Impl) for consistent evaluation. The complete list appears in Table 3.

Token Limits. Rather than hard limits, we employ soft constraints via prompts (2–3

Quality Validation. Two automated mechanisms ensure data integrity: (1) zero-token outputs trigger automatic retry, and (2) responses missing required JSON fields or violating constraints trigger re-prompting. Persistent failures after three attempts are flagged for manual review. Across 516 instances on 7 models, fewer than 3% required retry.

I Model Specifications

Table 5: Models used in our experiments

Model	Org.	API	Context	Param
GPT-4.1	OpenAI	OpenAI	1M	-
GPT-o3	OpenAI	OpenAI	1M	-
Gemini 1.5 Pro	Google	Vertex	128K	-
Qwen-3-32B	Alibaba	Groq	32K	32B
LLaMA-3.3-70B	Meta	Groq	128K	70B
Mistral Large	Mistral	Mistral	256K	675B
Deepseek-r1-0528	Deepseek	Deepinfra	128K	671B

"-" refers to undisclosed

J Manual Evaluation Protocol

To complement our automated metrics and better understand qualitative differences in feedback behavior across pipelines, we conducted a manual evaluation of a representative subset of model responses. This analysis focuses on pedagogical properties of feedback that are difficult to capture through accuracy-based metrics alone, such as scaffolding quality and answer revelation. Three domain experts independently annotated 10 problem instances per feedback pipeline (Student, Tutor, Teacher, and Judge), resulting in a total of 40 responses per model along four pedagogical dimensions. We use a minimally sufficient sampling strategy until inter-rater agreement exceeded a moderate Krippendorff's α threshold ($\alpha \geq 0.65$).

J.1 Annotators

Three expert annotators participated in the study. All annotators had at least two years of experience as teaching assistants for undergraduate Discrete Structures and had previously served as researchers for the logic tutoring system used to generate the dataset. This ensured familiarity with both propositional logic proof strategies and pedagogical objectives of step-level tutoring.

J.2 Sample Selection

We randomly sampled 10 problem instances per feedback pipeline (Tutor, Teacher, and Judge), resulting in a total of 30 annotated responses per model. Sampling includes a mix of correct, suboptimal, and incorrect student attempts. Following best practices for qualitative analysis in educational NLP, we used a minimally sufficient sampling strategy, increasing the sample size until inter-annotator agreement reached a stable, moderate level, Krippendorff's $\alpha \geq 0.70$ across all annotation dimensions.

J.3 Annotation Dimensions

Each feedback response was independently annotated along four pedagogically motivated dimensions:

Evaluation Rubrics. Feedback was annotated along four dimensions capturing both reasoning accuracy and pedagogical effectiveness:

- **Mistake Identification**: Whether the feedback correctly identifies errors or suboptimality in the student's proposed next step, including incorrect rule use, invalid derivations, or deviations from the optimal reasoning path.
- **Correctness**: Whether the feedback itself is logically sound and consistent with the correct solution, providing guidance that does not introduce new errors or misleading reasoning.
- **Answer Revelation**: Whether the feedback explicitly or implicitly reveals the correct next step, inference rule, or parent statements, thereby reducing the need for student reasoning.
- **Actionability**: Whether the feedback provides clear, concrete, and usable guidance that a student could plausibly act on to improve their next attempt.

Due to the cognitive complexity of step-level logic feedback and the time cost of expert annotation, we adopt a coarse 3-point ordinal scale (1–3) for all rubrics, with 1 = poor, 2 = partial, 3 = strong. This scale captures meaningful distinctions while maintaining annotation reliability and consistency.

J.4 Annotation Procedure

Annotators were provided with the problem context (givens, intermediates, and conclusion), the student's response, and the corresponding feedback generated by the pipeline under evaluation. Annotators worked independently and were blinded to model identity and quantitative performance results. Inter-annotator agreement was measured using Krippendorff's α, which is suitable for ordinal data and multiple annotators. Agreement exceeded a moderate reliability threshold across all four dimensions ($\alpha \geq 0.65$), indicating consistent interpretation of the rubric.

J.5 Analysis

Although qualitative and limited in scope, this analysis reveals consistent patterns that align with our quantitative results. Tutor feedback is typically vague but provides scaffolding, encouraging reasoning without explicit solution disclosure, whereas Teacher feedback often encodes the correct inference more directly (e.g., naming rules or premises). Judge feedback, when applied to noisy or overly revealing Teacher responses, tends to over-correct or abstract guidance, reducing instructional specificity, mirroring the conditional effectiveness of verification observed in our experiments.

K Design Implications for Education

Our findings have direct implications for the design of LLM-based tutoring systems in formal reasoning domains. First, verification should be conditionally deployed, not treated as a universal safeguard. Systems should estimate upstream tutor reliability and invoke verification primarily when error rates are high, rather than indiscriminately. Second, solution information access must be adaptive. Full derivational access expands coverage to harder states but increases the risk of premature revelation; lightweight access is preferable in low-complexity settings. Third, step-level reasoning alignment is critical: feedback that is locally valid but misaligned with the learner's current proof state can anchor incorrect reasoning and undermine learning. Finally, our results reveal a complexity ceiling beyond which no automated feedback strategy is reliable, suggesting the need for escalation strategies such as human intervention or extended symbolic reasoning support in highly complex cases.

L Future Directions

Several directions remain for future research. First, validating these findings with human learners is essential to assess learning gains, engagement, and transfer beyond LLM-based student simulations. Second, extending the framework to multi-turn tutoring dialogues may reveal different verification dynamics, including delayed correction or cumulative anchoring effects. Third, applying our approach to broader formal reasoning domains, such as first-order logic or program verification, would test its generality. Finally, developing adaptive routing mechanisms that dynamically select tutor roles, verification depth, and solution access based on student state and problem difficulty represents a promising direction toward robust, scalable educational AI systems.

Table 6: Effects of proof complexity and distance on correctness and feedback pipelines (Logistic Regression Coefficients, significance: *** $p < .001$, ** $p < .01$, * $p < .05$)

Model	Main Effects		Interaction with Pipeline		
	Complexity	Distance	Tutor × Comp.	Teacher × Comp.	Judge × Comp.
GPT-4.1	-0.35^{***}	-0.43^{***}	-0.02^{ns}	$+0.08^{ns}$	$+0.22^{***}$
GPT-o3	-0.42^{***}	-0.44^{***}	$+0.18^{*}$	$+0.13^{ns}$	$+0.24^{**}$
Gemini-1.5	-0.55^{***}	-0.37^{***}	$+0.55^{***}$	$+0.65^{***}$	$+0.61^{***}$
DeepSeek-R1	-0.36^{***}	-0.32^{***}	$+0.28^{***}$	$+0.36^{***}$	$+0.56^{***}$
Qwen-3	-0.53^{***}	-0.32^{***}	$+0.33^{***}$	$+0.37^{***}$	$+0.39^{***}$
Mistral	-0.42^{***}	-0.54^{***}	$+0.10^{ns}$	$+0.11^{ns}$	$+0.21^{*}$
LLaMA-3	-0.30^{***}	-0.49^{***}	-0.03^{ns}	-0.03^{ns}	$+0.04^{ns}$

Table 7: Feedback hurts student performance with corresponding mean complexity

Model	Tutor		Teacher		Judge	
	Count	Complexity	Count	Complexity	Count	Complexity
GPT	295	3.97	200	4.61	127	4.90
GPT-o3	267	4.14	200	4.64	126	5.23
Gemini	134	4.79	109	5.73	139	4.82
DeepSeek	121	4.74	81	6.12	104	5.08
LLaMA	383	3.44	332	3.60	307	3.60
Qwen	218	4.50	200	4.74	182	4.74
Mistral	319	3.84	302	3.94	226	4.23

Table 8: Tutor rule inference accuracy by rule type and model (%), best values are bolded across all rows

Rule	Actual	GPT-4.1	GPT-o3	Gemini	Mistral	Deepseek	Qwen	Llama-3.3
Simp	90	87	86	94	73	**96**	76	18
Add	61	81	77	78	62	85	**92**	23
MP	61	70	77	**96**	45	91	75	55
DS	53	43	45	81	60	77	**90**	17
MT	40	50	55	**82**	70	87	77	10
Impl	38	47	52	**86**	65	81	54	21
CP	36	30	33	83	10	**86**	58	17
DN	31	48	58	**83**	38	74	12	29
CD	27	7	11	**66**	3	**66**	55	11
DeM	21	47	38	**95**	66	85	66	14
HS	21	47	57	**100**	80	92	85	61
Conj	14	**92**	**92**	**92**	**92**	83	85	0
Comm	12	41	33	**72**	14	0	8	50
Equiv	7	0	0	33	**100**	**100**	0	0
Distr	4	**100**	**100**	**100**	0	0	50	50
Overall		58.3	60.1	**85.8**	54.8	**84.9**	70.2	24