Exploring the Design and Impact of Interactive Worked Examples for Learners with Varying Prior Knowledge

Sutapa Dey Tithi
Computer Science
North Carolina State University
Raleigh, North Carolina, USA
stithi@ncsu.edu

Xiaoyi Tian
Computer Science
North Carolina State University
Raleigh, North Carolina, USA
xtian9@ncsu.edu

Ally Limke
Computer Science
North Carolina State University
Raleigh, North Carolina, USA
anlimke@ncsu.edu

Min Chi
Computer Science
North Carolina State University
Raleigh, North Carolina, USA
mchi@ncsu.edu

Tiffany Barnes
Computer Science
North Carolina State University
Raleigh, North Carolina, USA
tmbarnes@ncsu.edu

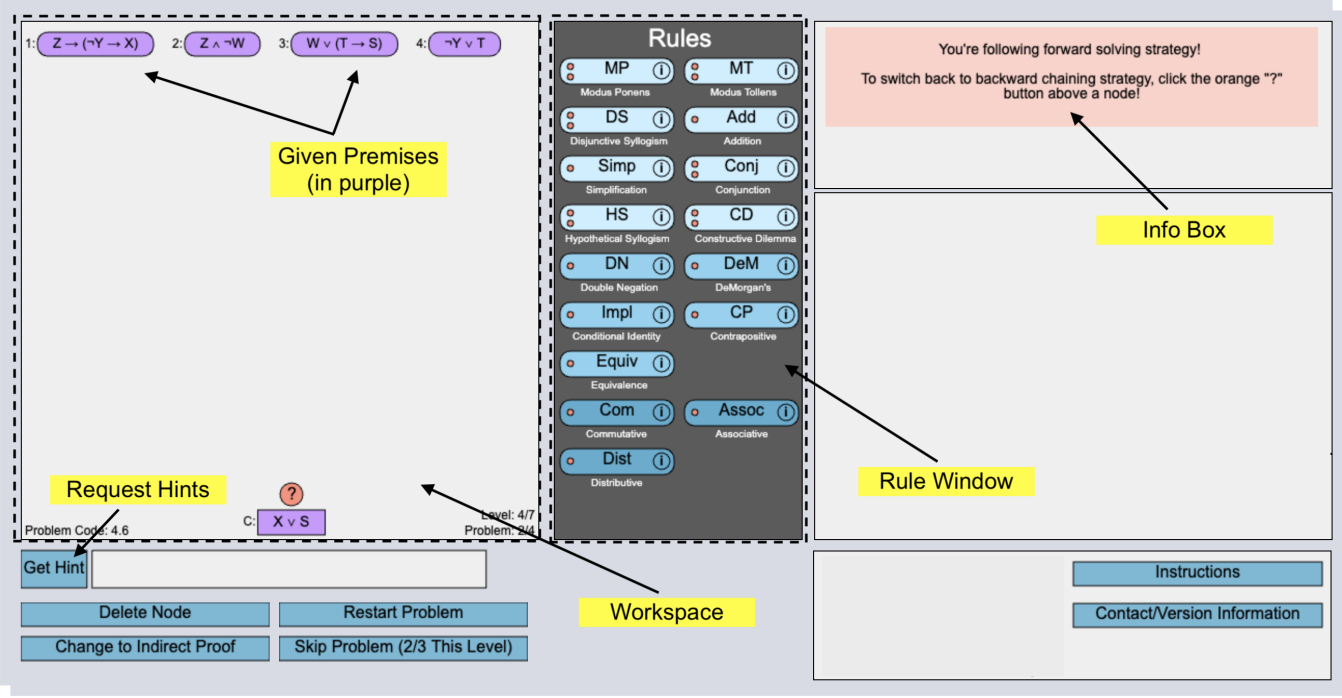


Figure 1: The tutor interface with the integrated three-panel design: student workspace (left), domain rules (center), and contextual information/instructions panel (right). The hint system and feedback message appear in the bottom-left corner.

Abstract

Tutoring systems improve learning through tailored interventions, such as worked examples, but often suffer from the aptitude-treatment interaction effect where low prior knowledge learners benefit more. We applied the ICAP learning theory to design two new types of worked examples, Buggy (students fix bugs), and Guided (students complete missing rules), requiring varying levels of cognitive engagement, and investigated their impact on learning in a controlled experiment with 155 undergraduate students in a logic problem solving tutor. Students in the Buggy and Guided examples groups performed significantly better on the posttest than those receiving passive worked examples. Buggy problems helped high prior knowledge learners whereas Guided problems helped low prior knowledge learners. Behavior analysis showed that Buggy produced more exploration-revision cycles, while Guided led to more help-seeking and fewer errors. This research contributes to the design of interventions in logic problem solving for varied levels of

This work is licensed under a Creative Commons Attribution 4.0 International License.

CHI '26, Barcelona, Spain

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3791631

learner knowledge and a novel application of behavior analysis to compare learner interactions with the tutor.

CCS Concepts

• **Human-centered computing → Empirical studies in interaction design**.

ACM Reference Format:
Sutapa Dey Tithi, Xiaoyi Tian, Ally Limke, Min Chi, and Tiffany Barnes. 2026. Exploring the Design and Impact of Interactive Worked Examples for Learners with Varying Prior Knowledge. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3772318.3791631

1 Introduction

Intelligent tutoring systems (ITSs) provide adaptive scaffolding and personalized interventions, demonstrating significant positive impacts on learning outcomes [44, 49]. However, the effectiveness of instructional interventions is not uniform across all learners. Research has found evidence of aptitude-treatment interaction (ATI) [18, 70], meaning that the effectiveness of specific instructional strategies varies significantly based on individual learner characteristics. These aptitude treatment interactions manifest across multiple dimensions of learner differences, including prior experience level, prior working memory, and incoming self-regulated learning ability [21, 36, 41, 83].

ITSs often incorporate two primary instructional approaches: Problem Solving (PS), where students independently construct solutions, or Worked Examples (WE), where the tutor presents step-by-step solutions for students to study. With regard to learners' abilities, research within ITSs has shown that high-ability learners can benefit from less guidance or more independent problem-solving practice. In contrast, low-ability learners benefit from more specific and direct guidance [3, 43]. Worked examples reduce the intrinsic cognitive load on learners and improve their learning [54]. Moreover, examples can save learners time without reducing their learning [48]. However, they may not be beneficial or could even be detrimental for learners with high prior knowledge, a phenomenon known as the expertise reversal effect [35, 51]. Thus, designing instructional interventions faces the challenge of creating opportunities so that learners of any incoming proficiency can master the same material and improve their learning outcomes in a similar amount of time.

The ICAP (Interactive, Constructive, Active, Passive) framework provides a theoretical lens for addressing this challenge [15]. This theory links students' cognitive engagement to their learning outcomes. It states that learning outcomes systematically increase as cognitive engagement increases, from *Passive* (receiving information), to *Active* (manipulating materials), to *Constructive* (generative), to *Interactive* (co-generative). Traditional worked examples, requiring only passive observation, fall into the lowest level of this hierarchy. There remain critical gaps in understanding: (1) how to design worked examples that promote higher levels of cognitive engagement while maintaining their scaffolding benefits, (2) empirical evidence of how worked examples with different levels of cognitive engagement help learners.

To address these gaps, we designed and empirically evaluated two types of interactive worked examples within an intelligent logic tutoring system: Buggy (students identify and fix bugs in worked solutions, promoting active engagement through debugging and fixing), and Guided (students reconstruct missing parts in worked solutions with step-by-step hints, promoting active reconstruction of proofs). The tutor also provides a baseline *Passive* WE and a *Constructive* PS intervention. The student-tutor interaction design and desired cognitive engagement in Buggy and Guided examples follow the ICAP framework and aim to increase interaction with the tutor relative to passive WE while maintaining more structure than independent PS. Propositional logic proofs are open-ended with multiple valid solutions. Our Buggy examples present expert-designed erroneous solutions without marking the incorrect steps, requiring students to understand both local rule applications and global proof coherence to successfully debug and fix incorrect steps. Guided examples decompose the solution into manageable chunks or subgoals, providing step-specific hints to help reconstruct the proof. Both Buggy and Guided examples are *Interactive* WEs, since they provide immediate feedback on student work. These two interventions are novel in that intelligent problem solving environments are often custom-built to enable the tracking of student work, and rarely have the capability to provide erroneous examples or subgoal-focused, partially-worked guided examples.

We integrated these interactive worked examples (Buggy, Guided) in an intelligent logic tutor along with passive Worked Example (WE) and constructive Problem Solving (PS). We investigated the impact of these worked examples on students' learning of propositional logic problem solving in a controlled experiment with 155 undergraduate Computer Science (CS) students. Students in different conditions practiced interleaving PS with one of: WE, Buggy, or Guided examples. We recorded students' clickstream interactions in the tutor and analyzed the log data. We investigated students' performance in posttest problems after training with three different interventions and analyzed further to answer the following research questions:

- RQ1 (Learning Outcomes): How do worked examples targeting different engagement levels (i.e., passive WE, Buggy examples, and Guided examples) affect students' performance in posttest problems?
- RQ2 (Moderation by Prior Knowledge): How do these effects vary with students' prior knowledge?
- RQ3 (Problem Solving Behavior): How do these interventions impact students' problem solving behaviors during training?

The contributions of this work are: (1) two theory-driven novel interactive worked example interface scaffolding designs in propositional logic learning promoting higher cognitive engagement; (2) empirical evidence of differential effectiveness of worked example interventions based on prior knowledge, informing adaptive logic tutoring; (3) a Markov model based visualization to interpret problem-solving behavior patterns; (4) design implications for worked examples that support guided proof construction, evaluation and debugging practice.

2 Related Work

2.1 Instructional Support in Intelligent Tutoring Systems

Multiple approaches have been explored within intelligent tutoring systems (ITSs) to facilitate student learning [5, 85]. These include Problem Solving (PS), where students solve the problem themselves, and Worked Example (WE), where the problem is solved for them step-by-step. WEs have been shown to reduce intrinsic cognitive load and improve learning [54]. However, the research has shown that WEs have mixed efficacy; while they may not be beneficial to students with high prior knowledge when problems are structured [51], they have also been shown to be most effective when they provide instructional explanations or rationales for the solution steps [62].

This phenomenon can be explained by Aptitude-Treatment Interaction (ATI) theory, which states that optimal learning occurs when instructional approaches align with individual learner characteristics [18]. Aptitudes can be broadly defined as any individual difference that affects learning, and these aptitudes interact with instructional approaches in complex, often non-linear ways [70]. Interventions that benefit one type of learner may actually harm another's performance. Although instructional methods such as WEs benefit novices, they may not be helpful to students with higher expertise. As students develop expertise, they build mental schemas that allow them efficient information processing. When experts receive the same detailed guidance that helps novices, they must process information they already know, wasting cognitive resources and thus reducing performance [35, 52].

The Cognitive Load theory identifies three types of cognitive load: intrinsic load (inherent to the material and learner's expertise level); extraneous load (imposed by poor design or irrelevant tasks); and germane load (productive effort supporting schema construction) [74]. WEs are designed to reduce intrinsic cognitive load and improve learning efficiency for novices [35, 73]. As learners progress towards higher expertise, techniques like requiring self-explanations help transition from example study to independent problem solving [14, 62], and can help achieve lower order learning objectives (remember/understand/early apply) [9]. For higher order objectives (analyze/evaluate/create), more active problem solving and structured interventions are typically more effective than fully WEs [15].

2.2 ICAP Learning Framework

The ICAP learning framework differentiates cognitive engagement activities into four modes: Interactive, Constructive, Active, and Passive [15]. Passive engagement includes activities where learners only receive information from the instructional materials, such as reading texts or watching instructional videos. Active engagement includes activities where students manipulate some parts of the learning materials, by copying some of the problem solution steps [78], or choosing a justification from a menu of options [17], and so forth. Constructive engagement requires learners to generate additional externalized outputs beyond what was provided in the learning materials, such as, comparing and contrasting cases [68], asking questions [24], etc. Finally, interactive engagement involves collaborative knowledge construction such as explaining to each other through dialogue with peers, tutors, or human-AI collaboration with intelligent systems [65]. The framework states that as students become more engaged with the learning materials, from passive to active to constructive to interactive, their learning will increase. Chi and Wylie presented empirical evidence from multiple studies to validate their ICAP hypothesis [15]. Their first study examined all four engagement modes within materials science instruction, demonstrating improved learning of 8-10% with each progressive mode. Then they presented two other studies that implemented passive, active, and constructive modes in the domains of evolutionary biology and plate tectonics, finding similar patterns of enhanced learning with increased engagement. Finally, they presented another study with pairwise comparisons across note-taking, concept mapping, and self-explanation yielding similar observations.

The default training intervention in our intelligent tutor allows students to practice propositional logic proofs independently with PS or traditional WEs. Similarly to solving programming problems, solving logic proofs requires students to understand a system of domain principles or rules and to apply them in a sequence to achieve a goal. In our tutor, when students solve WEs, the tutor shows every step with explanations, and students click through to progress forward or revisit the previous steps. In PS, on the other hand, students themselves need to apply appropriate domain rules to construct new steps towards achieving the goal. In our tutor, we integrated two new types of worked examples: Buggy (students fix bugs in worked solutions), and Guided (students reconstruct missing parts in worked solutions presented in Parsons problem fashion [20]). Both mechanisms are designed to deepen engagement and potentially improve learning outcomes. We elaborate on our design of Buggy and Guided examples in Section 3.2.

In addition to traditional PS and WE interventions, prior studies explored the effectiveness of erroneous examples and Parsons problems, particularly in domains such as programming and arithmetic. Buggy or erroneous examples leverage common mistakes made by students that can promote deeper understanding [25, 64]. When augmented with incorrect examples, algebra tutors enhanced student learning compared to those with only correct examples [10]. Similarly, studies with the AdaptErrEx decimals tutor showed enhanced learning when incorrect examples targeted common misconceptions [1, 46]. However, effectiveness may depend heavily on prior knowledge—novices may lack the necessary schemas to identify errors, while advanced learners may benefit from the critical thinking required for debugging. Well-designed and higher-level feedback, instead of giving away the solution, can facilitate productive debugging behavior and help students understand their problems [72]. Moreover, visualization has been found to be a good means for novice program debugging [11]. However, most existing Buggy example research focuses on short, single-step skills (e.g., arithmetic, decimals) or debugging in programming, and very few studies examine Buggy interventions in complex, multi-step problem solving tasks such as propositional logic proof construction. In a Parsons problem, students rearrange jumbled solution steps or complete the missing parts [20]. Research in programming education shows that Parsons problems (rearranging and completing partially blank lines of code into a valid program) are an effective exercise interface for

teaching programming patterns, significantly improving overall code-writing abilities [38, 79]. Recent work has extended this approach to mathematical proofs, demonstrating reduced difficulty in proof construction [58]. Understanding high-level contextual significance [59] and subgoal labels [47] can help students solve Parsons problems and improve their learning outcomes. Data-driven, subgoal-oriented Parsons problems can enhance students' problem decomposition-recomposition skills in solving propositional logic proofs [69]. However, students struggle with Parsons problems when they first encounter this type of structured or chunked problem or when the connections among different parts of the problem are complex [69]. While worked and partially worked examples exist in prior ITS research, their adaptation to other complex problem solving areas remains underexplored. Our Guided examples specifically target problem decomposition-recomposition skills while providing step-specific hints, reducing cognitive load in a complex problem solving environment.

In this study, we investigate the design and the impact of Buggy examples and Guided examples along with traditional WE and PS in our tutor. Our contribution is a systematic ICAP-aligned design of three worked-example types—passive, active, and constructive using identical learning materials for each representation, enabling a controlled comparison of engagement modes. However, identifying the mode of a particular activity can be a non-trivial task [23]. Our traditional WEs help students learn new information with *passive* engagement, and our PS helps students practice logic proofs through independent proof *construction*. In our design, Guided examples fade away or remove some connections in worked solutions, and students apply domain principles to complete missing connections. This task requires them to have an understanding of the given solution as a whole to deduce the domain principles that need to be applied to the missing part of the solution. Thus, we classify our Guided examples as an *active* ICAP mode. On the contrary, Buggy examples present worked solutions with expert-designed bugs, and students identify the bugs and fix them by typing the correct element. Unlike prior buggy example implementations, our design targets multi-step propositional logic proof construction, where debugging requires understanding of both local rule applications and global proof coherence. The task of finding bugs requires students to assess the given solution and produce qualifications such as "correct" or "incorrect", and the task of correcting the bugs needs them to add new information to the solution; thus, we classify these tasks as *constructive* ICAP mode. However, as we mentioned previously, classifying the mode of a particular activity is not trivial; Guided or Buggy examples can also be considered *interactive* depending on the context, since students often rely on the tutor's feedback or hints during the proof construction or the debugging process.

The increasing capabilities of generative AI systems create additional challenges and opportunities in education. Classroom and lab studies show both benefits and challenges: while generative AI-powered assistants can provide conceptual scaffolding without revealing complete solutions [39, 75], novices often struggle describing their intent in prompts, evaluating the correctness of generated solutions, and editing prompts when the generated code is incorrect [50]. Recent studies also examined how LLM-generated programming hints of varying granularity support or disappoint novices [82]. While human-AI collaboration with generative AI systems can help students in their learning process [57], LLMs are prone to hallucinating content that can be misaligned with course materials [32]. Thus, students need skills in evaluating, correcting, and optimizing any given content during human-AI collaborative problem solving. This creates new demands for educational systems that develop higher-order thinking skills rather than just problem-solving from scratch. Logic problem solving is a multi-step and open-ended procedure. In this work, our interaction designs target enhancing students' logic proof evaluation, debugging, and reconstruction capabilities while respecting individual differences in learning abilities.

3 System Design

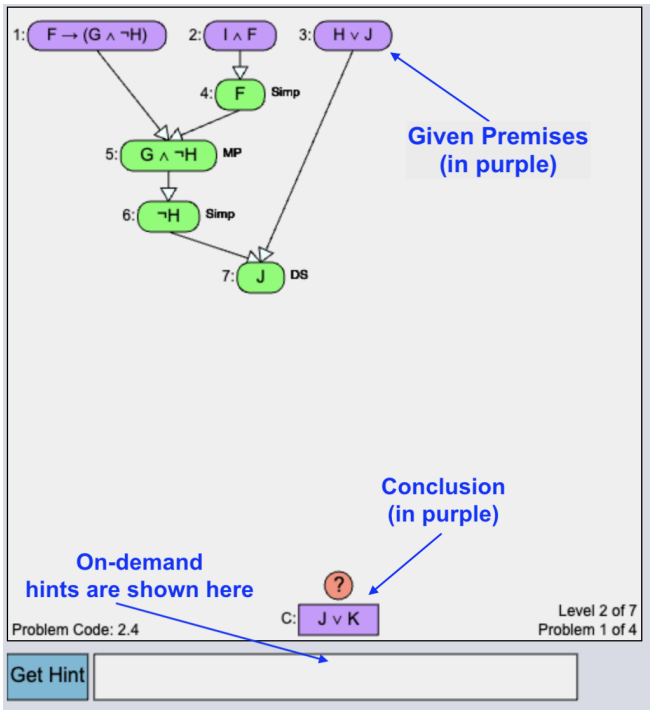
We designed two different instructional problem solving methods following the ICAP learning framework: Buggy examples (Buggy) and Guided examples (Guided). All methods use the same underlying propositional logic problems and require students to engage with the same set of domain principles/rules.

3.1 Tutor Overview

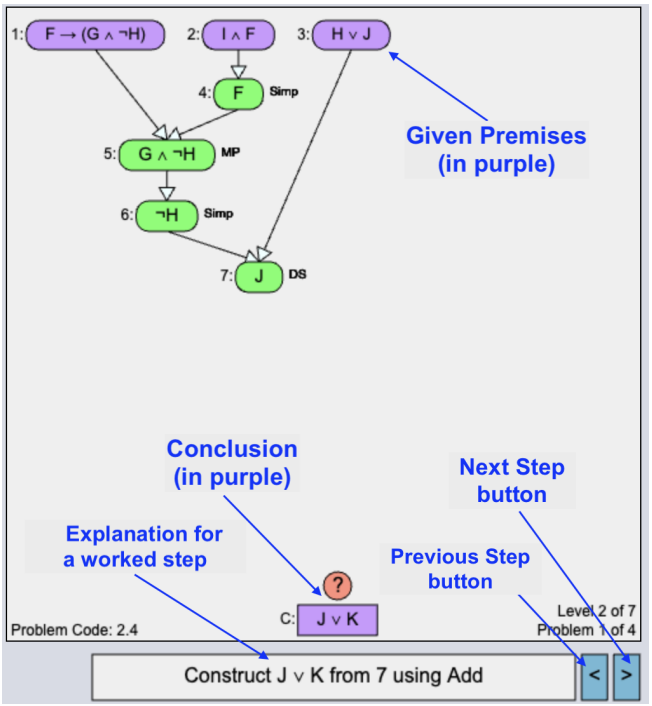
Our tutor is used in the context of an undergraduate Discrete Mathematics course where students practice open-ended, multi-step propositional logic problems. Students work within a three-panel interface as shown in Figure 1. The left panel provides the primary workspace where students construct proofs by deriving new logical statements, represented as nodes (circles). The center panel contains the domain rule buttons with their descriptions. Students use the rules to justify their derived statements; each node is justified using a rule and 1-2 parent nodes indicated by edges and a rule name label in the workspace. The right panel displays contextual information and instructions.

The proof construction process is presented in a graphical representation where each logical statement appears as a graphical node connected by edges that represent logical dependencies (as shown in Figure 2a). The given statements appear at the top in purple, while the target conclusion statement appears in purple at the bottom with a question mark, showing that it is not yet justified. Students construct proofs by deriving intermediate statements, creating a connected path from the givens to the conclusion. Each derivation step requires students to specify both the derived statement and the domain rule applied, and the parent statements supporting the derivation. This graphical representation allows students to visualize the overall structure of their proof.

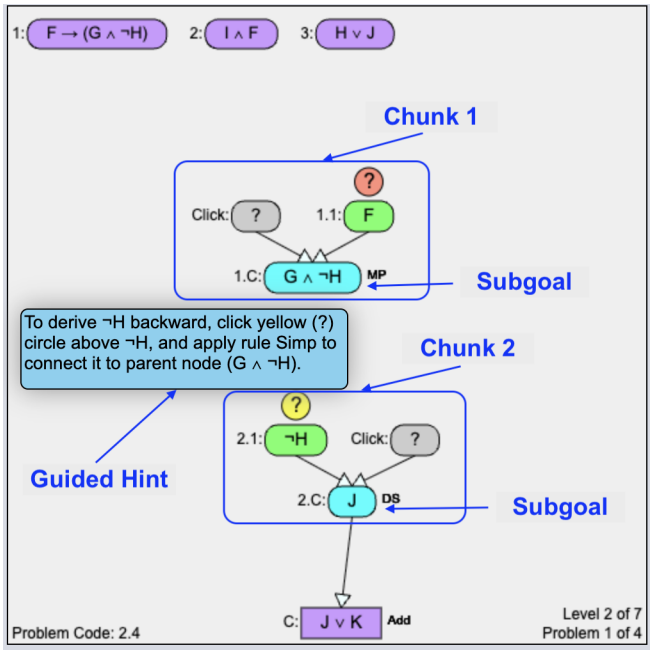
The tutor includes four sections: introduction, pretest, training, and posttest. The introduction consists of two problems where students get acquainted with the interface. Then, the students take the pretest, which consists of two problems. The pretest is used to measure the incoming proficiency of the students and assign them to intervention groups through stratified sampling. The training session then consists of five ordered levels of increasing difficulty, and each level consists of four problems. Finally, the posttest level (level 7) consists of six problems. During the pretest and posttest problems, the tutor does not provide hints for the next step, but offers immediate error messages when students make mistakes while deriving a new step. For each problem, the students receive a score between 0 and 100 based on efficient proof construction, with



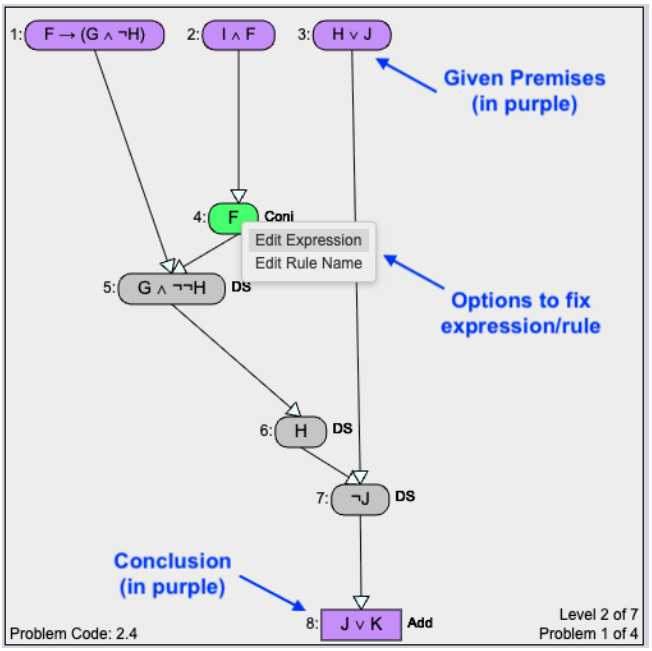
(a) Problem Solving (PS): Students independently derive all proof steps with minimal scaffolding



(b) Worked Example (WE): Complete solution demonstrated step-by-step



(c) Guided Example (Guided): Partial solution with missing connections



(d) Buggy Example (Buggy): Partial solution with bugs inserted

Figure 2: The tutor interfaces for four different problem types

higher scores corresponding to attempts with shorter solution size, higher accuracy of rule application, and shorter time.

Based on the intervention, the tutor presents four types of problems during training: Worked Example (WE), Problem Solving (PS), Guided Example (Guided), Buggy Example (Buggy).

3.2 Design ICAP Interventions

We integrated new problem types that progress in the ICAP framework, along with our existing problem types, PS and WE. Buggy example (Buggy) is designed to help students learn from active debugging and fixing, and Guided example (Guided) is designed to support active engagement by the construction of missing inference connections. Both problem types also support passive learning as they present an expert solution as a worked-out solution with some perturbations.

Problem Solving (PS) requires students to independently derive all proof steps and thus demands the highest cognitive engagement. Each problem solving step involves the same process: clicking on 1-2 parent nodes and a rule button, and entering the new derived statement, as shown in Figure 2a. The tutor verifies whether the parent nodes and the selected rule correctly justify the derived statement. Once a step is verified, a new derived node appears in the solution space. The new node is colored based on how often the same node was present in student solutions from previous semesters to this problem, where green means frequent, yellow means infrequent, and gray means never appeared. We call a node 'necessary' or 'needed' when its deletion would make a solution incomplete. If the tutor detects any incorrect attempt to derive a statement, it provides an error-specific feedback message to the student. PS problem types include a "Get Hint" button, which can be used to request a hint to progress in the solution. A student can get at most four on-demand hints in any training problem.

The design choice of PS reflects the goal of preserving independent problem solving while providing on-demand scaffolding as next-step hints. The frequency-based color coding of the nodes was generated by analyzing student solutions from previous years, and this can help students develop intuitions about common solution approaches without constraining their exploration.

Worked Example (WE) demonstrates a complete solution, performed by the tutor step by step as the students click on the next step (**>**) button (Figure 2b). Each step displays the derived statement, the justification rule, the parent statement(s), and indicates a high-level problem-solving operation (construct/extract/transform). The replay functionality allows students to review previous steps with their explanations, addressing a key limitation of traditional worked examples that proceed too quickly for novice understanding.

The interface maintains the same visual structure as PS but shifts the student's role from a constructor to an observer. Previously, our tutor supported these two problem types for practicing logic proofs, where PS falls into the 'Construction' category and WE falls into the 'Passive' category according to the ICAP framework.

Designing Buggy and Guided examples. In Buggy examples, we incorporated errors designed by an expert instructor with 20+ years experience in teaching Discrete Mathematics to change small elements of correct expert solutions. We address these errors as *bugs*. In the correct solutions, the bugs were inserted in two ways: (1) changing the logic operators in a correct statement, (2) changing the rule associated with a statement. The tutor provides contextual instructions about interacting with the interface to fix bugs. Initially, all the statements in the solution are gray, except the givens, which are purple and always correct. As shown in Figure 2d, after students click on any node, they can choose from two options: either to fix the statement or the rule associated with the node. Once they select an option, the tutor prompts with a text box, where students can type their answer. After submitting the answer, the tutor verifies its correctness, and identifies the step as correct/incorrect, or the tutor lets them know if that expression/rule was already correct. After the attempt, if the expression/rule is correct, it turns green.

Figure 2d shows a screenshot of an intermediate solution state of a Buggy example. The given nodes 1, 2, and 3 are shown at the top of the solution space, and the conclusion node 8 ($J \vee K$) is shown at the bottom. These purple nodes are always correct and cannot be modified. Node 4 (F) is green, implying that the student corrected the expression already. Now, if the student again clicks on node 4, the tutor prompts two options; the student may select the option 'Edit Rule Name' to replace the given rule ('Conj') with a correct rule ('Simp'). Thus, the student repeats the process of fixing until the solution is entirely correct. We hypothesize that students working with Buggy examples develop the critical evaluation skills of understanding and evaluating the correctness of solutions. This intervention spans through the *passive* (understanding of the given solution), *active* (finding the bugs), and *construct* (typing their own solution elements) learning activities in the ICAP framework.

In our tutor, Guided example (Guided) demonstrates a partially worked example with step-specific hints attached. In this problem, the proof structure is decomposed into chunks or *subgoals*, and each subgoal chunk groups related statements into logically meaningful units, aligning with Renkl's concept of 'meaningful building blocks' [63]. Additionally, we integrate step-specific hints into Guided examples to address the 'rationale gap' identified in traditional worked examples [61]. The hints guide students in choosing appropriate domain rules and completing the missing parts of the solution. Thus, Guided examples are designed to maintain low intrinsic load through subgoals and step-specific scaffolding while facilitating active problem solving participation. We hypothesize that students working with Guided examples will better learn the domain principles through an active solution reconstruction process. This intervention spans through the *passive* (understanding of the given solution), *active* (deducing the domain principles to complete the missing parts) learning activities in the ICAP framework.

Each Guided example provides students with all the statement nodes needed to complete a proof, but students must add a few justifications to connect all the nodes to one another with missing edges for rules. The nodes without incoming edges are unjustified. Guided examples guide students in justifying each unjustified node by specifying the rule used to derive it. In Figure 2c, statement 2.1: $-H$ can be derived from 1.C: $G \wedge -H$ using rule Simplification. To derive the statement, students are guided with a hint, as shown in the Figure 2c. To complete this step, students click on the yellow question mark above 2.1, choose the rule Simplification, and click on statement 1.C to show that there should be an edge from 1.C to 2.1. The solutions are divided into chunks, where important intermediate goals or subgoals in the problem are shown in light blue/cyan, grouped with the nodes used to derive them. For example, in this problem, there are 2 chunks 1 and 2, with subgoals 1.C and 2.C respectively, that are needed to complete the problem. The tutor guides students using popup hints with instructions to work backwards from the conclusion to connect to node 2.C, then connect 2.1 to chunk 1's conclusion 1.C, and then 1.1 to the givens.

Table 1: Example Activities in Different Problem Types: Problem Solving (PS), Worked Example (WE), Guided Example (Guided), Buggy Example (Buggy)

Student Action	Description	PS	WE	Guided	Buggy
Start Problem	Load problem interface and begin working on the assigned task	✓	✓	✓	✓
Read Rule	Hover over rule buttons to view descriptions and understand available logical operations	✓	✓	✓	✓
Hint Request	Click hint button (PS) or hover on the nodes (Guided) to receive tutor help about next steps	✓	–	✓	–
Correct Step	Correct application of a rule by selecting valid premises, appropriate rule, and correct derived statement	✓	✓	✓	✓
Incorrect Step	Attempt to perform a step with invalid premises, wrong rule, or incorrect derived statement	✓	✓	✓	✓
Delete Action	Remove previously entered node to backtrack and explore alternative solution paths	✓	–	–	–
Next Step	Navigate forward through WE steps to observe the next step in the solution	–	✓	–	–
Prev Step	Navigate backward through WE steps to review previously demonstrated steps in the solution	–	✓	–	–
End Problem	Complete the problem by successfully deriving the conclusion (PS), finishing example review (WE), connecting all statements (Guided), or fixing all errors (Buggy)	✓	✓	✓	✓

Table 1 summarizes the distinct student actions expected in each problem type.

4 User Study

4.1 Experimental Design

We conducted a controlled between-subjects study to evaluate the effectiveness of our developed interventions. We designed three training conditions in the logic tutor:

- Control (*Control*): Students received PS or WE (selected randomly) during training.
- Treatment 1 (*Buggy*): Students received PS or Buggy (selected randomly) during training.
- Treatment 2 (*Guided*): Students received PS or Guided (selected randomly) during training.

The system recorded all student interactions with the interface, including timestamp data, step-by-step action sequences, and help-seeking behaviors.

4.2 Participants

The tutor was deployed with 155 students in an undergraduate Discrete Mathematics course at a public research university in the United States in the Spring of 2025. We did not collect course-specific demographics; however, Discrete Math is a mandatory course for all CS majors. Therefore, for an approximation, we report the demographics of the 2021-22 graduating class of CS majors with the gender composition of 83% men and 17% women; and race/ethnicity of 58% white, 18.5% Asian, 3% Hispanic/Latin, 2% Black/African American, 9% other races, with the remaining 9.5% having international student status for whom race/ethnicity information was not available. This study was conducted at a university where we applied for and received an exemption for conducting the study under an IRB (Internal Review Board), and only authorized researchers could access the data collected from the participants.

Each participating student was assigned to one of the three training conditions after they completed the pretest problems. We used random stratified sampling, assigning students to groups after the pretest problems. They were assigned randomly while ensuring an even distribution of students with lower and higher pretest scores across all conditions implemented that semester. We compare only students who completed all 7 levels in the tutor, with 52 students in the *Control* group, 52 students in the *Buggy* group, and 51 students in the *Guided* group.

4.3 Performance Metrics

A student's problem score is a combination of normalized metrics for the **problem completion time**, **solution length**, and **rule application accuracy** on a single problem, which ranks a student based on how fast, efficient, and accurate they are. This composite metric captures multiple dimensions of problem solving skills. **Problem completion time** reflects procedural fluency, the ability to recognize and apply logical rules without excessive deliberation. Faster completion times suggest automated recognition of proof patterns, a key indicator of expertise development in formal reasoning tasks. **Solution length** indicates strategic thinking and optimal problem solving skills in proof construction, providing insight into students' proof-planning abilities and their tendency toward exploratory versus expert-like problem-solving approaches. Shorter solutions indicate better strategic planning. **Rule Application Accuracy** measures conceptual understanding through the ratio of correct rule applications to total application attempts.

The normalization ensures balanced weighting across metrics despite different scales and distributions. The resulting composite problem score helps us identify students who demonstrate balanced competency in speed, optimality, and accuracy, enabling targeted pedagogical interventions based on specific areas of weakness.

5 Study Results

We analyzed student **interaction logs** from three training intervention groups to measure their learning and performance.

5.1 RQ1: Impact on Learning Outcomes

To understand the impact of each of our training interventions on students' learning outcomes and performance, we analyzed students' score-based performance on test problems that they solved independently. Level 7 is the posttest section containing 6 test problems. To compare the performance across the three training conditions, we performed post hoc pairwise Mann-Whitney U tests to account for the non-normal nature of our data and problem-specific variability. We reported probability-based effect sizes (A) for this analysis as this measure is reported to be more robust when parametric assumptions are violated [16, 29, 53, 66]. This statistic represents the probability that a randomly selected participant from one group will have a higher value than a randomly selected participant from the other group. We provided A values with 95% confidence intervals (CI). Multiple comparisons were adjusted for using Bonferroni's correction $\alpha = 0.016$[1] [45, 80]. In addition to the statistical tests, we performed mixed-effects regression analysis [27, 81]. All student interactions with the tutor were recorded, thus there were no missing values in the dataset. Note that no significant

[1]In the pairwise tests, each data point was used in at most three tests: ($Control$, $Buggy$), ($Control$, $Guided$), and ($Buggy$, $Guided$). Thus, corrected $\alpha = 0.05/3 = 0.016$

Table 2: Overall Problem Score and Three Metrics: Rule Accuracy, Problem Completion Time (minutes), and Solution Length (Mean (SD)) in Test Problems. [Note: ** represents p-value<0.0003, * represents p-value<0.016, and † represents marginal significance (adjusted threshold after Bonferroni correction).]

Group (N)	Problem Score Mean (SD)	Rule Accuracy Mean (SD)	Time (minutes) Mean (SD)	Solution Length Mean (SD)
Control (52)	67.8 (22.2)	73.2 (22.1)	11.7 (22.6)	8.8 (3.8)
Buggy (51)	72.3 (21.8)*	76.8 (20.8)†	8.4 (14.5)*	8.7 (3.7)
Guided (52)	72.4 (22.4)*	79.2 (18.6)*	8.9 (16.1)*	8.7 (3.5)

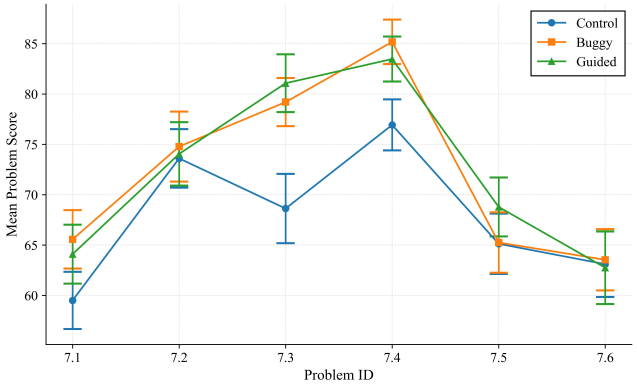


Figure 3: Comparisons of Problem Score Across Conditions during Posttest Problems in Level 7 [Note: Table 5 in Appendix A contains the same results for an alternative representation.]

differences were found in performance across the three groups in the pretest problems.

Problem Score. Figure 3 shows the problem score trajectories across all three conditions. No significant differences in pretest scores indicated equivalent initial competency levels across conditions.

Both treatment groups showed significantly higher posttest scores compared to the Control group (*mean* = 67.8). The *Buggy* group showed a significant improvement (*mean* = 72.3, $A = 0.56$, 95% CI [.52, .61], $p = .009$). Similarly, the *Guided* group also demonstrated a significant improvement (*mean* = 72.4, $A = 0.57$, 95% CI [.52, .61], $p = .004$) (see Table 2).

In addition to the statistical tests reported, we performed a mixed-effects regression analysis [27, 81] for all posttest problems to further verify the association between the given training treatment (*Control*, *Buggy*, *Guided*) and problem scores. We used problem IDs as a random effect variable to control for the impact of differences between test problems. We used the treatment type as the fixed effect and problem score as the dependent variable. The analysis, using the *Control* condition as the reference group, confirmed that training conditions had a significant overall effect on posttest scores. Compared to the *Control* condition (*mean* = 67.82, $SE = 3.32$), participants in the *Buggy* condition scored significantly higher ($\beta = 4.44, SE = 1.71, z = 2.60, p = .009$), as did participants in the *Guided* condition ($\beta = 4.56, SE = 1.70, z = 2.68, p = .007$). Specifically, the unstandardized coefficient (β) for *Buggy* indicates that participants in the *Buggy* group scored, on average, 4.44 points higher than the *Control* group. Similarly, the β for *Guided* shows an average score difference of 4.56 points compared to the *Control* group.

Rule Application Accuracy. In addition to the overall problem score, we observed students' step derivation behavior. Total correct steps are the number of correct rule applications in a problem. Total incorrect steps are the number of rule applications that are either incorrect by virtue of selecting a rule that does not apply to the arguments (e.g., using the rule *Simplification* on a logical statement that cannot be simplified) or not being able to correctly identify what statement a rule application would derive. Rule application accuracy is defined as the number of total correct rule applications divided by the total number of attempts for rule application. As shown in Table 2, *Guided* students achieved significantly higher rule application accuracy than *Control* in posttest problems (*Guided* (mean) = 79.3, *Control* (mean) = 73.2, $p = 0.001$, $A = .58$, 95% CI [.53, .62]). The rule application accuracy was not significantly different between *Buggy* and *Control* in posttest problems (*Buggy* (mean) = 76.8, *Control* (mean) = 73.2, $p = 0.04$, $A = .55$, 95% CI [.50, .60]).

Time. *Buggy* group had significantly lower problem completion time than the *Control* group in posttest problems (*Control* (mean) = 11.7 minutes, *Buggy* (mean) = 8.4 minutes, $p = 0.01$, $A = .56$, 95% CI [.51, .60]). Similarly, *Guided* group also had significantly lower problem completion time than the *Control* group in posttest problems (*Control* (mean) = 11.7 minutes, *Guided* (mean) = 8.9 minutes, $p = 0.001$, $A = .56$, 95% CI [.53, .63]). The problem completion time in the posttest problems for the groups *Buggy* and *Guided* was not significantly different ($p = 0.24$).

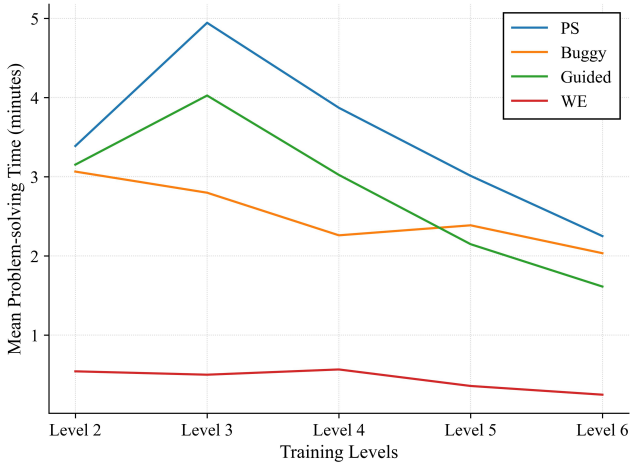


Figure 4: Comparisons of Mean Problem time for Each Problem Type during Training (Level 2 through Level 6)

As shown in Figure 4, PS required the longest completion times, reflecting its open-ended nature and higher cognitive load. In contrast, WE demanded minimal time investment, as students primarily engaged in passive observation (clicking buttons to progress through the tutor steps) rather than active proof construction. The mean problem completion time for Guided and Buggy is between; students spent more time on Guided examples in the earlier phase of training compared to the mean time spent on Buggy. In the later phases of the training (level 5 and level 6), students spent less time in Guided and more time in Buggy. Both treatment groups required more training time than the *Control* condition (*Buggy*: mean=1.12h, SD=0.5; *Guided*: mean=1.17h, SD=0.5; *Control*: mean=0.79h, SD=0.5), reflecting the additional cognitive engagement required for structured proof practice. Notably, despite spending more time during training, treatment groups achieved comparable total tutor times (around 4 hours across all conditions). More details appear in Table 6 (Appendix A).

Solution Length. As shown in Table 2, there were no significant differences in solution lengths across groups, indicating the treatments did not help students learn to construct shorter solutions.

5.2 RQ2: Interaction with Prior Knowledge

Prior research suggests that instructional interventions like worked examples and Parsons problems may have varying effectiveness based on learners' prior knowledge [35, 40]. To investigate this phenomenon in our context, we analyzed how the impact of Buggy and Guided examples differed based on prior knowledge. We categorized two prior knowledge groups (high and low) based on a median split on the pretest score. We compared metrics across two test sections (pretest and posttest) across high and low prior knowledge groups, using pairwise post hoc Mann-Whitney U tests with Bonferroni correction (corrected α = 0.05/3 or 0.016). Note that a total of 18 tests (2 pretest score groups, 3 dependent variables, and 3 pairwise tests for each group and performance metric) were carried out to compare the metrics presented in Table 3. Thus, a more conservative Bonferroni correction could be carried out to eliminate false positives. However, we do not want to introduce many false negatives while eliminating false positives. That is why, we decided the level of Bonferroni correction based on the number of unique pairwise tests that each data point participated in (3 pairwise tests across interventions), rather than on the number of related tests [8].

As shown in Table 3, the *Guided* group achieved significantly improved rule accuracy for low prior knowledge students at posttest problems (*Guided* (mean) = 78.7, *Control* (mean) = 71.8, p = 0.013, A = .59, 95% CI [.52, .65]). High prior knowledge students showed comparable posttest rule accuracy across groups *Control* and *Guided* (*Guided* (mean) = 79.7, *Control* (mean) = 74.4, p = 0.04, A = .56, 95% CI [.50, .62]), potentially suggesting a ceiling effect for learning rules. Appendix B includes the visualization of rule accuracy values in posttest problems across groups. Guided examples reduced problem completion time for low prior knowledge students during posttest problems (*Control* (mean) = 12.3 minutes, *Guided* (mean) = 8.6 minutes, p = 0.005, A = .60, 95% CI [.53, .66]). The solution lengths were not significantly different between the *Guided* and *Control* groups. These results suggest that Guided examples helped novices gain a better understanding of logic rules and complete the proof in less time.

The results for the group *Buggy* showed the opposite trend, revealing interesting aptitude-treatment interactions [70]. Across three dimensions (rule accuracy, problem time, solution length), there were no significant differences between groups *Buggy* and *Control* for the low prior knowledge students. In contrast, the high prior knowledge students in group *Buggy* achieved significantly higher rule application accuracy than their counterparts in the *Control* group in the posttest problems (*Buggy* (mean) = 82.6, *Control* (mean) = 74.4, p = 0.0004, A = .62, 95% CI [.55, .68]). High prior knowledge students in *Buggy* group achieved significant time reductions on posttest problems (*Control* (mean) = 11.2 minutes, *Buggy* (mean) = 6.3 minutes, p = 0.011, A = .58, 95% CI [.52, .65]). There was no significant difference in solution length for the high prior knowledge students between the two groups (*Control* (mean) = 8.8, *Buggy* (mean) = 8.2, p = 0.16, A = .55, 95% CI [.48, .61]). These results suggest that Buggy examples helped the advanced students further improve their understanding of logic rules and also complete their test problems in significantly less time.

In addition to the descriptive statistics presented in Table 3, we conducted mixed-effects regression analyses to further confirm how posttest performance metrics differed across training conditions after accounting for students' prior knowledge. We modeled Treatment and Prior Knowledge as fixed effects and student ID as a random effect variable. Although we present performance metric scores for the pretest in Table 3, no significant differences were found in performance metrics across the three groups in the pretest problems. The mixed-effects models confirmed differential effects across prior knowledge groups. For rule accuracy, low prior knowledge students in the *Guided* condition achieved significantly higher scores (β = 6.83, SE = 1.69, z = 4.0, $p < .001$) compared to *Control* (*mean* = 71.8), while *Buggy* examples showed no benefit (β = −0.72, SE = 2.30, z = −0.3, p = .75) for low prior knowledge group. Conversely, high prior knowledge students benefited significantly from *Buggy* examples (β = 8.94, SE = 3.93, z = 2.3, p = .023) compared to *Control* (*mean* = 74.4), implying a significant aptitude-treatment interaction. For problem completion time, compared to *Control* (*mean* = 12.3), *Guided* examples significantly reduced time for low prior knowledge students (β = −3.6, SE = 2.01, z = −2.0, p = .045), while *Buggy* examples significantly reduced time for high prior knowledge students (β = −4.9, SE = 2.36, z = −2.1, p = .03). For solution length, compared to *Control* (*mean* = 8.8), *Buggy* group had marginally shorter proofs for high prior knowledge students (β = −0.94, SE = 0.51, z = −1.8, p = 0.06), while no significant effects were found for the *Guided* group.

5.3 RQ3: Problem Solving Behavior Across Problem Types

The results from students' performance analysis showed that our new interventions with Guided and Buggy examples have the potential to improve students' learning performance. However, since they are a new type of problem-based training intervention, we acknowledged the necessity of analyzing their difficulty level compared to traditional training interventions such as PS or WE. Since intelligent tutoring systems are often used by students in the absence

Table 3: Performance Metrics (Mean (SD)) Across Three Training Conditions, Categorized on Pretest Scores. [Note: ** represents p-value<0.0003, * represents p-value<0.016, and † represents marginal significance (adjusted threshold after Bonferroni correction).]

Metric	Test	High Prior Knowledge			Low Prior Knowledge		
		Control $N = 26$	*Buggy* $N = 26$	*Guided* $N = 26$	*Control* $N = 26$	*Buggy* $N = 25$	*Guided* $N = 26$
Rule Accuracy	Pretest	64.1 (31.3)	61.4 (30.5)	63.5 (31.2)	44.2 (29.1)	39.7 (22.5)	45.1 (27.7)
	Posttest	74.4 (21.7)	**82.6 (18.5)***	79.7 (18.1)†	71.8 (22.4)	71.1 (21.5)	**78.7 (19.4)***
Problem Time (minutes)	Pretest	11.3 (14.3)	12.1 (16.7)	11.7 (18.1)	28.7 (28.3)	19.2 (18.7)	22.9 (21.4)
	Posttest	11.2 (23.1)	**6.3 (9.2)***	9.1 (14.7)	12.30 (22.2)	10.4 (18.1)	**8.6 (18.1)***
Solution Length	Pretest	4.9 (1.2)	5.0 (1.5)	4.9 (1.3)	5.5 (1.8)	6.1 (2.2)	7 (3.3)
	Posttest	8.8 (3.7)	8.2 (3.5)	8.4 (3.6)	8.74 (3.91)	9.1 (3.9)	9.1 (3.5)

of a human tutor, we investigated further the training difficulty in different problem types, aiming to avoid any increased difficulty so that the students can persist and learn. We conducted a comparative analysis and constructed first-order Markov models from student action sequences to navigate these problems [60]. The pattern of their actions provides insights into cognitive processes that traditional performance metrics may not capture. We generated all graph visualizations with Graphviz [22] and presented them in Figure 5.

We modeled student interactions as state transitions, computing $P(next\ action \mid current\ action)$ across three problem types: PS, Buggy, Guided. We do not present the analysis for WE problems, given that students could only observe the proof steps performed by the tutor. Figure 5 visualizes these transition diagrams. We displayed edges with >15% probability (preserving isolated nodes' strongest connections for graph connectivity) to balance detail with interpretability. We highlighted the transitions with probability > 30% in blue while fading out the rest. In the diagrams, "C Step" represents a correct attempt, and "Inc Step" represents an incorrect attempt in relevant problem types.

As shown in Figure 5a, Buggy examples demonstrate a distinctive "struggle pattern" characterized by tight loops on incorrect attempts and reading rule descriptions. Students frequently start with both correct attempts and incorrect attempts to fix any solution element, but they tend to remain stuck on incorrect attempts 51% of the time, while transitioning to correct attempts 27% of the time, and occasionally transitioning to reading rules. Similarly to incorrect attempts, once they start consulting the rules, they tend to read through the rules 70% of the time.

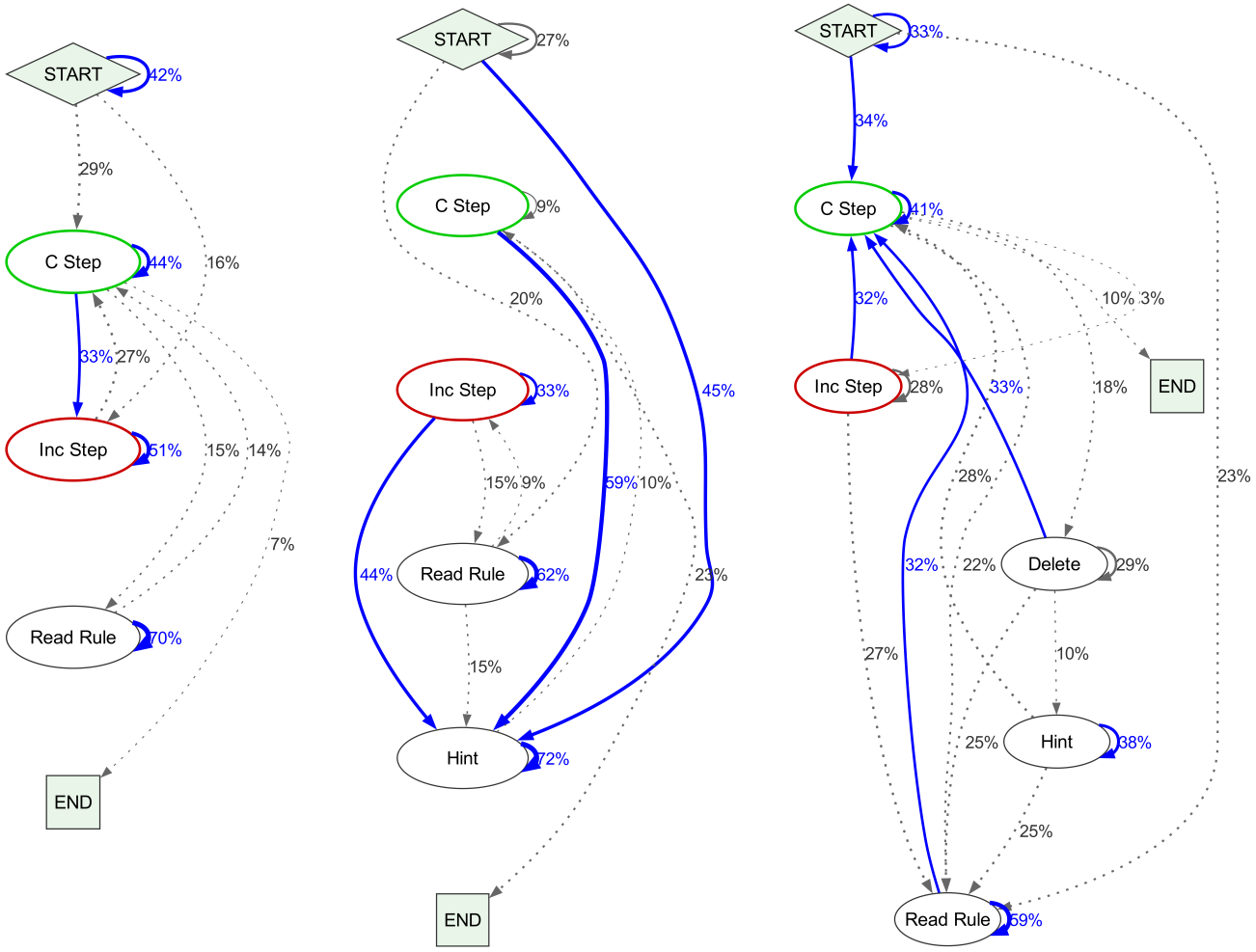
In contrast, Guided examples highlight the reliance on the subgoal hints. Incorrect attempts in Guided examples rarely lead to extended error cycles; students who make mistakes typically request hints (44% of the time) rather than persist with more incorrect attempts. Guided scaffolding helped redirect the struggling students toward help-seeking and correct attempts. However, the strong transitions to hints from all other actions (Start, Correct Step, Incorrect Step, and Read Rule) raise the question of over-reliance on the hints in Guided examples.

Problem Solving (PS) demonstrates a more diverse set of transition patterns. A prominent pattern in PS is the frequent Delete action, with 29% self-loops. The cycle (Correct Step, Delete, Hint, Read Rule, Correct Step) indicates frequent backtracking and revision behavior utilizing the hint system. Notably, other problem types are presented as a worked solution, so those problem representations do not support deleting given statements.

Transition patterns in Low vs. High prior knowledge groups. For each student, we generated transition probability values in each problem representation. We conducted an exploratory analysis to investigate the differences in transition patterns between students with low versus high prior knowledge in different problem representations. Due to the non-normal distribution of the data, we used Mann-Whitney U tests. Given the large number of possible transitions and the exploratory nature of this analysis, we do not apply Bonferroni-adjusted thresholds. In Table 4, we present the transition probabilities that differed between low and high prior knowledge groups with a p-value < 0.05 in the between-group Mann-Whitney U test and interpret them as suggestive patterns worthy of future investigation.

In the PS problems, low prior knowledge students were three times more likely to transition from hints to incorrect steps than high prior knowledge students (30.4% vs. 10.1%, $p = 0.009$, $A = .91$, 95% CI [.73, 1.0]). They also showed a stronger tendency to request hints at problem start (67.6% vs. 50.1%, $p = 0.06$, $A = .66$, 95% CI [.49, .81]), suggesting their uncertainty about how to start the proof at the beginning. Both groups demonstrated deletion patterns, though high prior knowledge students showed slightly higher multiple deletions (43.7% vs. 37.4%, $p = 0.06$, $A = .60$, 95% CI [.50, .70]).

In the Buggy examples, high prior knowledge students demonstrated significantly higher rates of consecutive correct steps (51.2% vs. 44.2%, $p = 0.001$, $A = .60$, 95% CI [.54, .66]) and more successful transitions from reading rules to correct actions (41.8% vs. 32.6%, $p = 0.02$, $A = .59$, 95% CI [.52, .67]). They were also more likely to begin problems with correct steps (73.7% vs. 64.7%, $p = 0.03$, $A = .58$, 95% CI [.51, .65]), while low-knowledge students showed higher rates of transitioning to errors (44.4% vs. 39.6% for correct to incorrect transitions, $p = 0.04$, $A = .57$, 95% CI [.50, .63]).



(a) Buggy Example (Buggy) **(b) Guided Example (Guided)** **(c) Problem Solving (PS)**

Figure 5: Transition Diagrams of Student Actions Across Different Problem Types

In the Guided examples, high prior knowledge students showed better error recovery, with 54.9% of incorrect steps followed by correct ones compared to only 36.3% for low prior knowledge students ($p = 0.04$, $A = .64$, 95% CI [.51, .76]). They also more frequently read rules after errors (58.8% vs. 45.8%, $p = 0.05$, $A = .72$, 95% CI [.50, .73]), suggesting more strategic help-seeking behavior. These patterns indicate that prior knowledge may also influence the strategic use of available support systems across different instructional interventions.

6 Discussion

This paper examined two different worked examples (Buggy, Guided) for learning propositional logic problem solving in an intelligent tutoring system. The results demonstrate that the choice between scaffolded practice (Guided) and debugging (Buggy) approaches has implications for both learning outcomes and problem solving strategies, with effects strongly moderated by students' prior knowledge.

6.1 Buggy examples benefit advanced learners while Guided examples support novices.

The interventions with Buggy and Guided examples had differential effectiveness across proficiency levels. High prior knowledge students benefited more from Buggy examples, achieving 82.6% rule accuracy compared to 74.4% for the *Control* group. One factor behind this differential benefit can be the higher demand for cognitive engagement and critical thinking in Buggy examples, as a result of the limited tutor help apart from the given partial solution. In Buggy examples, students must simultaneously hold correct logical principles in mind while identifying and correcting bugs. This metacognitive reasoning exercise—distinguishing

Table 4: Transition Patterns in Low vs. High prior knowledge students Across Problem Types. The reported Low/High Prior Knowledge % values are $100 \times P(\textbf{next action} \mid \textbf{current action})$. Effect size A denotes the probability that a randomly selected participant from low group will have a higher value than a randomly selected participant from the high group. A is asymmetric ($A_{\textbf{low,high}} = 1 - A_{\textbf{high,low}}$).

Problem Type	Transition	Low Prior (%)	High Prior (%)	p-value	A, 95% CI
PS	Hint → Inc Step	30.4	10.1	.009	.91, [.73, 1.0]
	C Step → Delete	26.2	21.8	.04	.58, [.50, .66]
	Delete → Delete	37.4	43.7	.06	.40, [.30, .50]
	Start → Hint	67.6	50.1	.06	.66, [.49, .81]
Buggy	C Step → C Step	44.2	51.2	.001	.40, [.34, .46]
	Read Rule → C Step	32.6	41.8	.02	.41, [.33, .48]
	Start → C Step	64.7	73.7	.03	.42, [.35, .49]
	C Step → Inc Step	44.4	39.6	.04	.57, [.50, .63]
Guided	Inc Step → C Step	36.3	54.9	.04	.36, [.24, .49]
	Inc Step → Read Rule	45.8	58.8	.05	.38, [.27, .50]

correct from incorrect—may strengthen the conceptual understanding for those who already have high prior knowledge. Research on desirable difficulties has also suggested that students can gain long-term learning benefits from more challenging materials, as long as they have sufficient knowledge or support to overcome the challenge [7]. Our *Buggy* intervention proved ineffective for low prior knowledge students, who showed no improvement over *Control*. Without stable mental models of correct reasoning, these students may struggle to recognize bugs or may even reinforce misconceptions through exposure to Buggy examples.

In contrast, Guided examples demonstrated more universal benefits, particularly for low prior knowledge students who achieved 78.7% rule application accuracy in posttest problems. In Guided examples, the tutor presents a partial solution and offers step-by-step hints to complete the missing parts. Thus, the scaffolded nature of Guided examples, while requiring active participation in completing missing parts in a given solution, creates an optimal challenge level for novices. Grouping related statements into subgoal clusters may also help novices in schema construction. By offloading the construction of new statements to the tutor, students can focus their limited working memory on understanding logical relationships.

Design recommendation. These findings validate that learning activities in higher-order ICAP modes require higher cognitive management. Systems should dynamically adjust interface constraints based on inferred expertise: providing structured scaffolding to novices to manage cognitive load, while introducing "productive friction" (like debugging tasks) to more proficient or expert-like users to prevent disengagement. While previous research has used AI to select activities adaptively [2, 6, 55] and to track students' knowledge states [13, 30, 31, 84], we recommend that designers leverage real-time granular interaction logs to predict the users' current knowledge state and cognitive requirements. For instance, systems could detect when a user is overwhelmed (requiring scaffolding) or coasting (requiring more challenge) and adapt the interaction mode accordingly.

6.2 Debugging requires productive struggle, while guided proof reconstruction leads to less struggle.

The behavioral patterns extracted from our Markov models demonstrate why these differential effects emerge. Students working on Buggy examples exhibited a productive struggle pattern of repeated attempts to identify and fix errors. Prior research demonstrated the "productive failure" paradigm—engaging students in solving complex problems without the provision of support structures can promote learning [37]. High prior knowledge students in *Buggy* group showed improved scores in posttest problems, indicating that they benefited from engaging with Buggy examples.

Guided examples, conversely, produced minimal error loops and higher help-seeking. Students who made mistakes typically requested hints (44% probability) rather than persisting with more incorrect attempts. Research within ITSs has shown that low ability students benefit from more specific and direct guidance [3]. The guided structure and step-specific hints in the Guided examples helped struggling students toward correct step derivation. These results align with results from a previous study where low prior knowledge learners achieved significantly better rule accuracy and problem solving times, and their thematic analysis showed that students felt the Parsons problems with hints simplified the problems and reduced cognitive load through guided workflows [77]. This scaffolded nature may be particularly beneficial for novice learners, helping build confidence alongside competence.

Notably, both treatment groups spent approximately 40% more time during training than the *Control* group yet achieved faster posttest completion times. Students who engaged with either debugging activities in Buggy examples or proof reconstruction in Guided examples developed a higher accuracy of rule application than those alternating between Problem Solving (PS) and Worked Example (WE). This finding suggests that the cognitive effort required by the interventions, while initially costly, yields superior transfer performance.

Design recommendation. Research in complex search tasks has demonstrated that analyzing *nonlinear state transition patterns* allows systems to predict implicit task states, effectively distinguishing between productive exploration and struggle, without relying solely on final outcomes [42]. Transition probabilities provide immediate, high-fidelity signals of a user's confusion or confidence, offering a richer dataset than delayed performance scores. The observed struggle cycles found in our Markov analysis must be flagged in real-time as an unproductive interaction, prompting an immediate change in the instructional state. This applies not only to tutoring but also to any iterative task environment, where preventing failure loops is crucial, such as code debugging environments that could interrupt excessive "trial-and-error" attempts with contextual explanations and documentation links [67], or complex data entry workflows that prevent users from repeating the same formatting error.

6.3 Each problem type shapes distinct cognitive strategies through interface design.

The distinct transition patterns across problem types reveal how interface design shapes cognitive strategies. The deletion-heavy patterns in PS—particularly for low-knowledge students—may indicate *guess-and-check* strategies instead of genuine refinement strategies. Understanding these deletion patterns could inform more targeted interventions, such as evaluative feedback on deletions or metacognitive prompts about strategy selection. Our interventions—*Buggy* and *Guided*—each also created unique behavioral patterns: *Buggy* demonstrated persistent error correction attempts, while *Guided* had fewer errors and higher use of hints, indicating a potential for "gaming" behaviors or over-reliance on tutor help.

Design recommendation. The effectiveness of the Buggy (detection/repair) and Guided (completion/structure) examples stemmed from their highly distinct interfaces, which guided students toward different cognitive goals. Interfaces should, therefore, explicitly align with their specific pedagogical or domain-specific intent. The interaction design must clearly communicate the type of thinking required, effectively scaffolding the user's mental model. For instance, this approach has been employed in broader human-AI co-creation by Suh et al., whose system *Sensecape* distinguishes between "exploration" and "sensemaking" [71]. Their interface provides separate, distinct views for divergent ideation and convergent structuring, thus helping users maintain their high-level structural intent without being overwhelmed by low-level generative content.

7 Limitations and Future Directions

Although our tutor maintains the same learning materials across different problem representations, requiring varying levels of cognitive engagement, several design limitations emerged that could be addressed in future work.

One significant limitation of our study is the exclusive reliance on quantitative interaction log data. While these logs allowed us to analyze specific behaviors and performance, we lack the crucial qualitative context necessary for a complete understanding of the learning process. Specifically, we do not have data regarding students' moment-to-moment affective states (e.g., frustration, confidence), their specific study environments, or detailed background characteristics beyond the clickstream data from their interaction with the tutor. This limits our ability to fully interpret the why behind the aptitude-treatment interactions—for example, whether low prior knowledge students benefited from Guided examples due to reduced confusion or increased motivation.

The *availability of scaffolding* could be more balanced across interventions; students could request hints in PS and Guided examples, but not in Buggy examples. This was a design choice to increase the cognitive challenge in Buggy examples, but the Markov model analysis revealed substantial struggle patterns when students encountered these problems. Similar to Guided examples, future iterations should introduce hints in Buggy examples when students show unproductive struggle. Moreover, prior research suggests that self-reflection and the process of *making sense of errors* is potentially a more productive learning process [10, 25]. Buggy examples can be augmented with self-explanations, which may yield a better understanding of the sources of the difficulty students face in these problems.

Our results suggested that Buggy examples benefited high prior knowledge students, while Guided examples were more effective for low prior knowledge students. This finding suggests the need for *adaptive problem selection* based on learner expertise. It should be noted that learners' expertise evolves throughout training; that is why a static assignment based on the pretest scores may be insufficient. Future work should implement adaptive interventions that continuously track students' knowledge growth and provide *personalized* Guided examples or Buggy examples to help students improve their weak areas. Generative AI has shown potential to generate personalized content and hints [19, 26, 33, 34, 56]. Integrating generative AI in tutoring systems has introduced concerns as well [4, 12, 28]. Recent work evaluating LLMs for logic tutoring found that while LLMs achieved up to 86.7% accuracy in logic proof construction, their hint explanations often lacked pedagogical appropriateness and failed to explain the high-level rationale behind hints [76]. LLMs may generate buggy examples containing arbitrary bugs rather than pedagogically meaningful error patterns found in actual students' solutions. In Guided examples, LLMs may provide overly detailed hints that could hinder the desirable difficulty that our interventions aimed to create. Thus, leveraging generative AI in tutoring systems like ours requires a careful design of human-AI interaction and collaboration, with access to real-time student models and human-in-the-loop review, to ensure pedagogical quality. The ICAP framework can make human-AI collaborative problem solving more valuable, and help designers focus on improving interactive aspects of AI that probe and challenge human thinking to achieve the highest benefits for humans and for problem solving tasks.

Another limitation is that, in our Markov model analysis, we only presented the transition probabilities, rather than also including the actual frequency values of each action and/or transition. Within the scope of this paper, we only presented an exploratory analysis of such fine-grained data. Future works can further investigate this data to determine ways to differentiate productive and unproductive struggle or identify scaffolding needs. Moreover, we experimented with a single tutor for a short period of time. Longer studies, such as across a semester, with multiple tutors, could generalize our findings and also reveal if the benefits from the training are transferable across domains over time.

It is noteworthy that the observed effects in our study may change depending on factors such as differences in task type or domain complexity. Our study was conducted in a logic tutor, with problems that are considered to be open-ended but well-structured, meaning that there are many solutions but they use defined rules. We anticipate that observed benefits might be most pronounced for tasks in well-structured problem-solving domains, such as programming, math, and science. Our findings may not fully generalize to less structured domains (e.g., writing or design tasks), where the "correctness" of a step or bug is less clear. Logic proofs in our tutor usually have 12 steps. For simpler tasks, the cognitive engagement benefits of Buggy and Guided examples may diminish, as students can fix errors or complete missing connections through straightforward pattern matching rather than engaging in conceptual reasoning. On the other hand, in a highly complex problem

solving domain, Buggy examples may overwhelm learners with excessive cognitive load. The expertise reversal effect we observed suggests that *overall task complexity should scale with learner expertise.*

8 Conclusion

In this study, we augmented a propositional logic tutor with two new types of worked examples, Buggy and Guided examples, and investigated their impacts on learning compared to passive WEs. The results show that the Buggy and Guided groups performed significantly better on the posttest problems than the Control group. Buggy problems especially helped high prior knowledge learners, whereas guided problems helped low prior knowledge learners, improving rule application accuracy and problem completion time. Through a novel application of behavior analysis, we further investigated student problem-solving attempts: students in the Buggy group required more persistent effort to get out of the loop of incorrect attempts. In contrast, students in the Guided group had fewer errors and higher help-seeking from the tutor. These contrasting patterns demonstrate how different ICAP engagement modes can be strategically designed to match students' varying prior knowledge. These findings contribute to the growing body of work on AI-teacher and AI-student tutoring interactions and provide empirical evidence that ICAP theory can guide practical design decisions in intelligent tutoring systems, and for complex domains like propositional logic where learners arrive with varying prior knowledge. Rather than simply making learning materials "more interactive," designers must consider the interaction between engagement mode and learner expertise. Designers of human-AI collaborative problem solving systems should further consider designing opportunities for challenge and assistance that can scaffold the productive struggle needed for human learning– rather than propagating systems that promote over-reliance. These principles likely extend to other online learning environments with high variability in users' prior experience, and demonstrate how ICAP can be used to guide the design of complex human-AI collaborative systems for improved problem solving.

9 Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and Grammarly to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

Acknowledgments

The work is supported by National Science Foundation (NSF) grant 2013502.

References

[1] Deanne M Adams, Bruce M McLaren, Kelley Durkin, Richard E Mayer, Bethany Rittle-Johnson, Seiji Isotani, and Martin Van Velsen. 2014. Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior* 36 (2014), 401–411.

[2] Nazia Alam, Behrooz Mostafavi, S_D Tithi, Min Chi, and Tiffany Barnes. 2024. How much training is needed? Reducing training time using deep reinforcement learning in an intelligent tutor. Proceedings of the 17th International Conference on Educational Data Mining ….

[3] Ivon Arroyo, Joseph E Beck, Beverly Park Woolf, Carole R Beal, and Klaus Schultz. 2000. Macroadapting Animalwatch to gender and cognitive differences with respect to hint interactivity and symbolism. In *International conference on intelligent tutoring systems*. Springer, 574–583.

[4] Mia Magdalena Bangerl, Leonie Disch, Tamara David, and Viktoria Pammer-Schindler. 2025. CreAItive Collaboration? Users' Misjudgment of AI-Creativity Affects Their Collaborative Performance. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–17.

[5] 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*. Springer, 373–382.

[6] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. 2020. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–12.

[7] Robert A Bjork and Elizabeth L Bjork. 2020. Desirable difficulties in theory and practice. *Journal of Applied research in Memory and Cognition* 9, 4 (2020), 475.

[8] J Martin Bland and Douglas G Altman. 1995. Multiple significance tests: the Bonferroni method. *Bmj* 310, 6973 (1995), 170.

[9] Benjamin Samuel Bloom. 2010. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

[10] Julie L Booth, Karin E Lange, Kenneth R Koedinger, and Kristie J Newton. 2013. Using example problems to improve student learning in algebra: Differentiating between correct and incorrect examples. *Learning and Instruction* 25 (2013), 24–34.

[11] Peter Brusilovsky. 1993. Program visualization as a debugging tool for novices. In *INTERACT'93 and CHI'93 Conference Companion on Human Factors in Computing Systems*. 29–30.

[12] Xinyue Chen, Kunlin Ruan, Kexin Phyllis Ju, Nathan Yap, and Xu Wang. 2025. More ai assistance reduces cognitive engagement: Examining the ai assistance dilemma in ai-supported note-taking. *Proceedings of the ACM on Human-Computer Interaction* 9, 7 (2025), 1–29.

[13] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2011. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction* 21, 1 (2011), 137–180.

[14] Michelene TH Chi, Nicholas De Leeuw, Mei-Hung Chiu, and Christian LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive science* 18, 3 (1994), 439–477.

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

[16] Robert Coe. 2002. It's the effect size, stupid. In *British educational research association annual conference*, Vol. 12. 14.

[17] Cristina Conati and Kurt Vanlehn. 2000. Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education* 11 (2000), 389–415.

[18] Lee Joseph Cronbach and Richard E Snow. 1981. *Aptitudes and instructional methods: A handbook for research on interactions*. Ardent Media.

[19] Tiffany D Do, Usama Bin Shafqat, Elsie Ling, and Nikhil Sarda. 2025. PAIGE: Examining learning outcomes and experiences with personalized AI-generated educational podcasts. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–12.

[20] Barbara J Ericson, Paul Denny, James Prather, Rodrigo Duran, Arto Hellas, Juho Leinonen, Craig S Miller, Briana B Morrison, Janice L Pearce, and Susan H Rodger. 2022. Parsons problems and beyond: Systematic literature review and empirical study designs. *Proceedings of the 2022 working group reports on innovation and technology in Computer Science education* (2022), 191–234.

[21] Douglas Fuchs, Devin M Kearns, Lynn S Fuchs, Amy M Elleman, Jennifer K Gilbert, Samuel Patton, Peng Peng, and Donald L Compton. 2019. Using moderator analysis to identify the first-grade children who benefit more and less from a reading comprehension program: A step toward aptitude-by-treatment interaction. *Exceptional children* 85, 2 (2019), 229–247.

[22] Emden R Gansner and Stephen C North. 2000. An open graph visualization system and its applications to software engineering. *Software: practice and experience* 30, 11 (2000), 1203–1233.

[23] Adam M Gaweda and Collin F Lynch. 2021. Student Practice Sessions Modeled as ICAP Activity Silos. *International Educational Data Mining Society* (2021).

[24] Arthur C Graesser and Natalie K Person. 1994. Question asking during tutoring. *American educational research journal* 31, 1 (1994), 104–137.

[25] Cornelia S Große and Alexander Renkl. 2007. Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and instruction* 17, 6 (2007), 612–634.

[26] Quan Connie Gu, Daniel Hickey, and Kimiko Ryokai. 2025. When AI Tells Their Story: Researchers' Reactions to AI-Generated Podcasts as a Tool for Communicating Research. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–6.

[27] Xavier A Harrison, Lynda Donaldson, Maria Eugenia Correa-Cano, Julian Evans, David N Fisher, Cecily ED Goodwin, Beth S Robinson, David J Hodgson, and Richard Inger. 2018. A brief introduction to mixed effects modelling and multi-model inference in ecology. *PeerJ* 6 (2018), e4794.

[28] Emma Harvey, Allison Koenecke, and Rene F Kizilcec. 2025. " Don't Forget the Teachers": Towards an Educator-Centered Understanding of Harms from Large Language Models in Education. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–19.

[29] Carl C Haynes and Barbara J Ericson. 2021. Problem-solving efficiency and cognitive load for adaptive parsons problems vs. writing the equivalent code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[30] Md Mirajul Islam, Xi Yang, Rajesh Debnath, Adittya Shoukarjya Saha, and Min Chi. 2025. A Generalized Apprenticeship Learning Framework for Capturing Evolving Student Pedagogical Strategies. In *International Conference on Artificial Intelligence in Education*. Springer, 393–408.

[31] Md Mirajul Islam, Xi Yang, John Hostetter, Adittya Soukarjya Saha, and Min Chi. 2024. A generalized apprenticeship learning framework for modeling heterogeneous student pedagogical strategies. *arXiv preprint arXiv:2406.02450* (2024).

[32] 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*. 491–499.

[33] Hyoungwook Jin, Seonghee Lee, Hyungyu Shin, and Juho Kim. 2024. Teach ai how to code: Using large language models as teachable agents for programming education. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–28.

[34] Hyoungwook Jin, Minju Yoo, Jeongeon Park, Yokyung Lee, Xu Wang, and Juho Kim. 2025. Teachtune: Reviewing pedagogical agents against diverse student profiles with simulated students. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–28.

[35] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing cognitive load in adaptive multimedia learning*. IGI Global Scientific Publishing, 58–80.

[36] Slava Kalyuga, Paul Chandler, Juhani Tuovinen, and John Sweller. 2001. When problem solving is superior to studying worked examples. *Journal of educational psychology* 93, 3 (2001), 579.

[37] Manu Kapur. 2016. Examining productive failure, productive success, unproductive failure, and unproductive success in learning. *Educational Psychologist* 51, 2 (2016), 289–299.

[38] Ville Karavirta, Juha Helminen, and Petri Ihantola. 2012. A mobile learning application for parsons problems with automatic feedback. In *Proceedings of the 12th koli calling international conference on computing education research*. 11–18.

[39] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. In *Proceedings of the 2024 chi conference on human factors in computing systems*. 1–20.

[40] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science* 36, 5 (2012), 757–798.

[41] Janina Lehmann, Christina Goussios, and Tina Seufert. 2016. Working memory capacity and disfluency effect: An aptitude-treatment-interaction study. *Metacognition and Learning* 11, 1 (2016), 89–105.

[42] Jiqun Liu, Shawon Sarkar, and Chirag Shah. 2020. Identifying and Predicting the States of Complex Search Tasks. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval* (Vancouver BC, Canada) *(CHIIR '20)*. Association for Computing Machinery, New York, NY, USA, 193–202. doi:10.1145/3343413.3377976

[43] Rosemary Luckin, Benedict Du Boulay, et al. 1999. Ecolab: The development and evaluation of a Vygotskian design framework. *International journal of artificial Intelligence in Education* 10, 2 (1999), 198–220.

[44] Wenting Ma, Olusola O Adesope, John C Nesbit, and Qing Liu. 2014. Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of educational psychology* 106, 4 (2014), 901.

[45] Patrick E McKnight and Julius Najab. 2010. Mann-whitney U test. *The Corsini encyclopedia of psychology* (2010), 1–1.

[46] Bruce M McLaren, Deanne M Adams, and Richard E Mayer. 2015. Delayed learning effects with erroneous examples: a study of learning decimals with a web-based tutor. *International Journal of Artificial Intelligence in Education* 25, 4 (2015), 520–542.

[47] Briana B Morrison, Lauren E Margulieux, Barbara Ericson, and Mark Guzdial. 2016. Subgoals help students solve Parsons problems. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 42–47.

[48] Behrooz Mostafavi, Guojing Zhou, Collin Lynch, Min Chi, and Tiffany Barnes. 2015. Data-driven worked examples improve retention and completion in a logic tutor. In *International conference on artificial intelligence in education*. Springer, 726–729.

[49] Tom Murray. [n. d.]. An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art. *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software* ([n. d.]), 491–544.

[50] Sydney Nguyen, Hannah McLean Babe, Yangtian Zi, Arjun Guha, Carolyn Jane Anderson, and Molly Q Feldman. 2024. How beginning programmers and code llms (mis) read each other. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–26.

[51] Fleurie Nievelstein, Tamara Van Gog, Gijs Van Dijck, and Henny PA Boshuizen. 2013. The worked example and expertise reversal effect in less structured tasks: Learning to reason about legal cases. *Contemporary Educational Psychology* 38, 2 (2013), 118–125.

[52] Annishka Oksa, Slava Kalyuga, and Paul Chandler. 2010. Expertise reversal effect in using explanatory notes for readers of Shakespearean text. *Instructional Science* 38, 3 (2010), 217–236.

[53] Anna-Marie Ortloff, Florin Martius, Mischa Meier, Theo Raimbault, Lisa Geierhaas, and Matthew Smith. 2025. Small, Medium, Large? A Meta-Study of Effect Sizes at CHI to Aid Interpretation of Effect Sizes and Power Calculation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–28.

[54] Fred Paas, Alexander Renkl, and John Sweller. 2003. Cognitive load theory and instructional design: Recent developments. *Educational psychologist* 38, 1 (2003), 1–4.

[55] Zachary A Pardos, Matthew Tang, Ioannis Anastasopoulos, Shreya K Sheel, and Ethan Zhang. 2023. Oatutor: An open-source adaptive tutoring system and curated content library for learning sciences research. In *Proceedings of the 2023 chi conference on human factors in computing systems*. 1–17.

[56] Minju Park, Sojung Kim, Seunghyun Lee, Soonwoo Kwon, and Kyuseok Kim. 2024. Empowering personalized learning through a conversation-based tutoring system with student modeling. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–10.

[57] Ivica Pesovski, Ricardo Santos, Roberto Henriques, and Vladimir Trajkovik. 2024. Generative AI for customizable learning experiences. *Sustainability* 16, 7 (2024), 3034.

[58] Seth Poulsen, Mahesh Viswanathan, Geoffrey L Herman, and Matthew West. 2022. Evaluating proof blocks problems as exam questions. *ACM Inroads* 13, 1 (2022), 41–51.

[59] James Prather, John Homer, Paul Denny, Brett A Becker, John Marsden, and Garrett Powell. 2022. Scaffolding Task Planning Using Abstract Parsons Problems. In *IFIP World Conference on Computers in Education*. Springer, 591–602.

[60] Lawrence R Rabiner. 2002. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (2002), 257–286.

[61] Alexander Renkl. 2002. Worked-out examples: Instructional explanations support learning by self-explanations. *Learning and instruction* 12, 5 (2002), 529–556.

[62] Alexander Renkl. 2005. The worked-out-example principle in multimedia learning. *The Cambridge handbook of multimedia learning* 1 (2005), 229–245.

[63] Alexander Renkl, Robert K Atkinson, and Uwe H Maier. 2000. From studying examples to solving problems: Fading worked-out solution steps helps learning. In *Proceeding of the 22nd Annual Conference of the Cognitive Science Society*. 393–398.

[64] J Elizabeth Richey, Juan Miguel L Andres-Bray, Michael Mogessie, Richard Scruggs, Juliana MAL Andres, Jon R Star, Ryan S Baker, and Bruce M McLaren. 2019. More confusion and frustration, better learning: The impact of erroneous examples. *Computers & Education* 139 (2019), 173–190.

[65] Rod D Roscoe and Michelene TH Chi. 2007. Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of educational research* 77, 4 (2007), 534–574.

[66] John Ruscio. 2008. A probability-based measure of effect size: robustness to base rates and other factors. *Psychological methods* 13, 1 (2008), 19.

[67] Eldon Schoop, Forrest Huang, and Bjoern Hartmann. 2021. UMLAUT: Debugging Deep Learning Programs using Program Structure and Model Behavior. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 310, 16 pages. doi:10.1145/3411764.3445538

[68] Daniel L Schwartz and John D Bransford. 1998. A time for telling. *Cognition and instruction* 16, 4 (1998), 475–5223.

[69] Preya Shabrina, Behrooz Mostafavi, Sutapa Dey Tithi, Min Chi, and Tiffany Barnes. 2023. Learning Problem Decomposition-Recomposition with Data-Driven Chunky Parsons Problems within an Intelligent Logic Tutor. *International Educational Data Mining Society* (2023).

[70] Richard E Snow. 1991. Aptitude-treatment interaction as a framework for research on individual differences in psychotherapy. *Journal of consulting and clinical psychology* 59, 2 (1991), 205.

[71] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling Multilevel Exploration and Sensemaking with Large Language Models. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 1, 18 pages. doi:10.1145/3586183.3606756

Table 5: Mean Problem Scores Posttest Problems (Level 7) Across Conditions

Problem ID	Control	Buggy	Guided
	Mean (SD)	Mean (SD)	Mean (SD)
7.1	59.50 (2.84)	65.57 (2.90)	64.10 (2.93)
7.2	73.61 (2.91)	74.78 (3.47)	74.05 (3.15)
7.3	68.63 (3.44)	79.20 (2.40)	81.07 (2.87)
7.4	76.93 (2.53)	85.18 (2.21)	83.47 (2.23)
7.5	65.12 (2.99)	65.25 (3.00)	68.78 (2.92)
7.6	63.10 (3.25)	63.54 (3.05)	62.75 (3.61)

[72] Ryo Suzuki, Gustavo Soares, Elena Glassman, Andrew Head, Loris D'Antoni, and Björn Hartmann. 2017. Exploring the design space of automatically synthesized hints for introductory programming assignments. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2951–2958.

[73] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[74] John Sweller. 2011. Cognitive load theory. In *Psychology of learning and motivation*. Vol. 55. Elsevier, 37–76.

[75] Benyamin Tabarsi, Aditya Basarkar, Xukun Liu, Dongkuan DK Xu, and Tiffany Barnes. 2025. MerryQuery: A Trustworthy LLM-Powered Tool Providing Personalized Support for Educators and Students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 29700–29702.

[76] Sutapa Dey Tithi, Arun Kumar Ramesh, Clara DiMarco, Xiaoyi Tian, Nazia Alam, Kimia Fazeli, and Tiffany Barnes. 2025. The Promise and Limits of LLMs in Constructing Proofs and Hints for Logic Problems in Intelligent Tutoring Systems. *Computers and Education: Artificial Intelligence* 9 (2025), 100490. doi:10.1016/j.caeai.2025.100490

[77] Sutapa Dey Tithi, Xiaoyi Tian, Min Chi, and Tiffany Barnes. 2025. Investigating the Impact and Student Perceptions of Guided Parsons Problems for Learning Logic with Subgoals. *arXiv preprint arXiv:2505.04712* (2025).

[78] Kurt VanLehn, Arthur C Graesser, G Tanner Jackson, Pamela Jordan, Andrew Olney, and Carolyn P Rosé. 2007. When are tutorial dialogues more effective than reading? *Cognitive science* 31, 1 (2007), 3–62.

[79] Nathaniel Weinman, Armando Fox, and Marti A Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–4.

[80] Eric W Weisstein. 2004. Bonferroni correction. *https://mathworld. wolfram. com/* (2004).

[81] Brady T West, Kathleen B Welch, and Andrzej T Galecki. 2022. *Linear mixed models: a practical guide using statistical software*. Chapman and Hall/CRC.

[82] Ruiwei Xiao, Xinying Hou, and John Stamper. 2024. Exploring how multiple levels of GPT-generated programming hints support or disappoint novices. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–10.

[83] Yu-chu Yeh and Chun Fu Lin. 2015. Aptitude-treatment interactions during creativity training in e-learning: How meaning-making, self-regulation, and knowledge management influence creativity. *Journal of Educational Technology & Society* 18, 1 (2015), 119–131.

[84] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*. Springer, 171–180.

[85] Rui Zhi, Thomas W Price, Samiha Marwan, Alexandra Milliken, Tiffany Barnes, and Min Chi. 2019. Exploring the impact of worked examples in a novice programming environment. In *Proceedings of the 50th acm technical symposium on computer science education*. 98–104.

A Appendix A

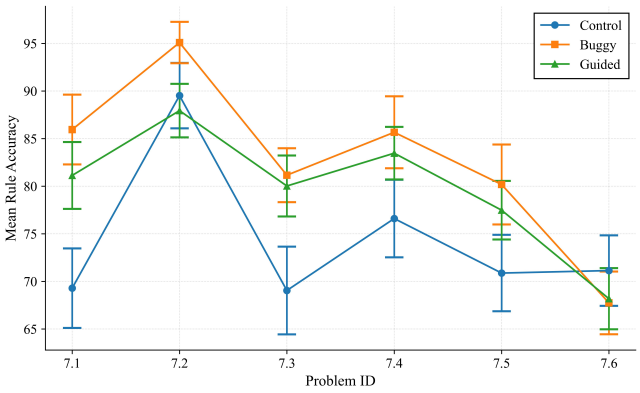
Table 5 is added for accessibility, and contains the same information as in Figure 3.

Table 6 presents the time spent on different sections of the tutor by each student on average.

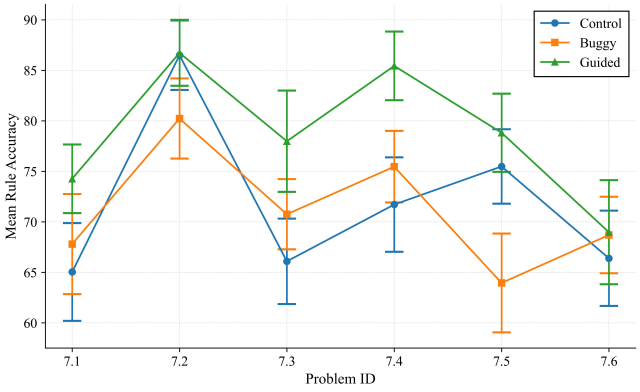
Table 6: Time comparison across the three conditions. All values are hours (Mean (SD)).

Section	*Control*	*Buggy*	*Guided*
Training	0.79 (0.5)	1.12 (0.5)	1.17 (0.5)
Level-End test	1.62 (0.9)	1.56 (0.8)	1.43 (1.1)
Posttest	1.17 (0.8)	0.84 (0.5)	0.89 (0.5)
Total Tutor	4.24 (1.8)	4.05 (1.43)	4.03 (1.8)

B Appendix B



(a) Upper Median



(b) Lower Median

Figure 6: Comparisons of Rule Accuracy Across Conditions during Posttest Problems.