Experiences in Reverse Engineering Student Mistakes Using Generative AI

Dr. Maria R Ebling, United States Military Academy

Maria R. Ebling is an Assistant Professor at the United States Military Academy at West Point in the Department of Electrical Engineering and Computer Science. She earned the Ph.D. and M.S. degrees in Computer Science from Carnegie Mellon University and the B.S. in Mathematics with a Computer Science option from Harvey Mudd College. Her research interests lie at the intersection of distributed systems and human-computer interaction, especially around pervasive computing, and the Internet of Things. Prior to joining the faculty of West Point, Dr. Ebling spent three years as the Chief Technology Officer at Medaptive Health, a small health-tech startup. Before that, she spent nearly 20 years at the IBM T. J. Watson Research Center, in various roles ranging from research staff member to director and was a member of the IBM Academy of Technology. Dr. Ebling is an ACM Distinguished Scientist. She served as the Editor-in-Chief of IEEE Pervasive Computing from 2014-2017 and continues to serve the publication as a member of the Advisory Board. Over the years, she has served on numerous program committees in mobile computing.

Mr. Geoffrey Moores, United States Military Academy

MAJ. Geoff Moores is an instructor in the Department of Electrical Engineering & Computer Science at the United States Military Academy (USMA) in West Point, New York. He is an active duty Cyber Officer with a masters degree in Computer Science from UMD. His research interests include network science, optimization algorithms, and computer science education.

Can Generative AI Identify Student Misconceptions in CPU Scheduling?

Abstract

The capabilities of generative artificial intelligence to enhance higher education represent a significant unknown for many instructors. In this paper, we present our experience using large language models (LLMs) to determine what misconceptions 89 undergraduate students have based on their incorrect answers to a four-part exam problem on CPU scheduling policies. First, we manually analyzed each of the 120 incorrect solutions to identify underlying misconceptions and random mistakes. We then asked three production-quality LLMs (ChatGPT, Claude, and Gemini) to do the same task and compared their responses to ours. The LLM responses varied widely in quality and accuracy. The LLMs tended to perform better when students made systematic errors and on simpler policies. Interestingly, the LLMs tended to perform better when the instructors initially disagreed on the misconception and the error was systematic. Overall, our analysis suggests using LLMs as a supplementary diagnostic tool for student errors, rather than a standalone substitute for instructor feedback.

Introduction

Students make mistakes. Determining the misconception underlying those mistakes is frequently nontrivial and time consuming, but allows instructors to correct their students' misconceptions more readily. Our goal in conducting this research is to better identify the misconceptions underlying mistakes made by students learning CPU scheduling. We sought to use that newfound knowledge to improve the feedback provided by an open-source CPU Scheduling simulation[1]. For our data set, we analyzed student answers to a CPU scheduling exam question. Our analysis revealed a diverse range of errors. As a result, we decided to explore the use of generative artificial intelligence[1] (GenAI) to determine student misconceptions.

We note this task is significantly more complex than identifying errors at the surface level. In this work, we wanted to identify what *mistaken belief* could reasonably lead to the errors evident in a given solution. We evaluated how often the LLMs' explanations of student misconceptions aligned with those of instructors. In cases where the LLM's explanation did not align with that of the instructors, we examined whether it represented a *plausible* alternative – meaning it offered a different, yet logically consistent, interpretation of the student's error. Finally, we examined whether LLMs could explain errors that instructors could not.

[1]We use the term GenAI to refer to the field and LLM to refer to the models.

Given our original goal of improving the feedback provided by a CPU scheduling simulation, we considered the topic of single-processor CPU scheduling in a typical undergraduate Operating Systems course. We analyzed student answers to an exam question on this topic across a cohort of two separate offerings of the course. The question sets up the scheduling problem and asks students to show the order in which processes would be scheduled according to different scheduling policies. We coded the student answers into a string, identified the unique incorrect answers, and asked multiple LLMs to explain the students' misconceptions. We then evaluated the LLMs' responses to identify their agreement with the instructor assessment and explanatory value provided.

This paper is organized as follows. First, we provide the context for this work and differentiate it from prior efforts. Next, we outline our method for analyzing the CPU scheduling data. We then summarize the results received from the various LLMs. We follow the results with a reflection on our experiences, a discussion of the limitations and ideas for future work, and finally conclude the paper.

Background

Understanding student misconceptions is a central challenge in education at large. Cavalcanti et al.[2] conducted a systematic literature review of automated feedback in online learning, finding common approaches such as answer comparison and progress dashboards. Although some use of natural language processing was noted, most tools were subject specific and predated the availability of LLMs.

Contemporary alternatives to GenAI for modeling student misconceptions rely on structured frameworks. For example, the In-Context Teaching model [3] uses Bayesian updates over input-output examples to infer a student's internal model of a concept, such as fractional arithmetic. This allows the generation of targeted interventions based on the most likely areas of misunderstanding, but is computationally complex and requires significant data for each student. A different technique, programming-by-demonstration [4], generates explainable programs to model student logic, but suffers from high implementation overhead and brittleness. In contrast to these approaches, we explored the use of GenAI's powerful LLMs.

Other recent efforts to evaluate student work include explorations of clustering of student answers in *embedding spaces* which leverage supervised learning models to generate high-dimensional vectors of inputs. These vectors and their spatial relationships can capture useful semantic meanings [5]. Shi et al. [6] used clusters of embedding vectors of incorrect code submissions in programming tasks to reveal latent patterns of misunderstanding. Similarly, clustering techniques applied to open-ended responses in physics education [7] have shown that AI-derived groupings can mirror expert annotations. Although helpful as a preprocessing step to better understand similarities and differences between student responses, clustering methods leave the significant burden of analysis of misconceptions to the human analyst. In contrast to this work, we apply GenAI to the task of determining misconceptions with the goal of reducing this burden.

Most recently, there is a growing body of research into using GenAI to provide feedback to help students learn specific topics. For example, Neshaei et al. used generative AI to provide feedback to students learning argumentation[8] whereas Verhelst and Belpaeme explored the use of LLMs

in helping students learning a foreign language [9]. Both of these examples fall more clearly into topic areas that artificial intelligence has focused on solving – language, reasoning, and speech. In contrast, our focus is on more specialized knowledge and targets identifying misconceptions on topics the LLMs have not been directly trained to do.

Another thread of investigation uses LLMs to provide students with feedback. Nguyen et al. compared the feedback from LLMs to that from instructors and found that OpenAI's GPT-4 provided feedback comparable to that of instructors[10]. Balse et al. investigated OpenAI's GPT-3's ability to provide feedback on programming assessments and found a wide range of scores in terms of accuracy of checking code correctness, critiquing code, and suggesting appropriate changes[11]. Like Balse et al., we found that LLMs produced feedback that varied substantially in terms of accuracy and correctness.

Researchers have also begun to integrate LLMs into intelligent tutoring systems. Mueller et al. showed the value of historical context in such an integration when used with introductory programming students[12]. Liu and M'hiri explored the use of a virtual TA based on OpenAI's GPT-3.5 in an introductory programming course [13], finding that the virtual TA still required human oversight. Our work supports their conclusion that LLMs require human oversight and extends that conclusion to the problem of diagnosing student misconceptions to a more advanced course.

Pedagogical Chain-of-Thought (PedCoT) prompting [14] represents a recent advance in the use of LLMs to identify misconceptions. PedCoT demonstrates that LLMs can better identify student errors when guided to identify pedagogical components of the problem, such as applicable concepts (remember), relevant problem solving methods (understand), and the execution or computation steps needed (apply). In the form presented in their paper, student work must be pre-processed into discrete steps and fed into a two-prompt sequence for full analysis. Although their approach seemed promising, we found better initial results by directing the LLM's chain-of-thought to identify specific scheduling errors that supported the student misconception hypothesis within a single prompt.

Despite these innovations, using generative AI to interpret student misconceptions in procedural techniques has only been explored in limited ways. This paper addresses that gap by sharing our experiences in understanding the extent to which current LLMs can both identify student errors and trace them back to specific misconceptions.

Method

Using exam questions from our institution's Operating Systems course, we identify mistakes that students make on questions about CPU scheduling. We then query three LLMs about what misconception is illustrated by the mistake. This study was reviewed by our institution.

Exam Question

The key question for CPU scheduling gave students a set of 5 processes, A through E, as well as their start and service times. The start times, in order, were 0, 2, 2, 5, and 11; the service times were 10, 5, 7, 3, and 1. In the event that multiple processes had the same start time (e.g.,

processes B and C), students were told to handle them in alphabetical order by process name (e.g., B then C). The question asked them to identify which process runs at each time unit using four different scheduling policies: first-in-first-out (FIFO), shortest-time-to-completion-first (STCF), round robin with a time quanta of 2 (RRq2), and multi-level feedback queue (MLFQ) with three queues that have increasing time quanta (Q_0 with 1 unit, Q_1 with 2 units, and Q_2 with 4 units). These policies are commonly covered in standard undergraduate textbooks[15] and in undergraduate Operating Systems courses. Students respond to each part of the question by completing a two-dimensional grid.

To support this experiment, we coded the student answers to each part of the question as a string with the alphabet $\Sigma = \{A, B, C, D, E\}$ such that if process C ran at time 1 and process E ran at time 2, the string representing the student's answer would begin with "CE". We then analyzed student answers to find the unique incorrect answers that students made. We manually examined these unique incorrect answers to identify what mistake we thought the student(s) had made. In our analysis, we identified whether we thought the mistake was systematic (meaning the student followed a procedure, but the wrong one) or random (most likely caused by a bookkeeping error), or both. Both authors independently made a determination, with disagreements resolved by discussion.

Next, both annotators independently generated explanations for the misconception behind each unique, incorrect answer. We then assessed agreement by comparing explanations for semantic equivalence. For example, on a RRq2 problem, one instructor stated "Pre-empted B to serve D, jumping D to front of the queue; did same with E" while the other stated "New arrivals are immediately scheduled / interrupt". After discussion the instructors agreed that the student was immediately scheduling new arrivals and preempting running processes. On an MLFQ problem, one instructor diagnosed the student's error as "Round robin'ing with increasing quanta; allows new processes to jump the queue" whereas the other instructor diagnosed the same error as "First in first out on any queue". After discussion, the consensus was that the student was using round robin with increasing quanta and allowing new processes to jump the queue.

The initial agreement varied based on problem type, generally decreasing based on complexity: FIFO=100% (n=2); STCF=82% (n=11); RRq2=65% (n=20); MLFQ=58% (n=31). We resolved discrepancies through discussion to produce a final consensus explanation. In the cases where there was initial disagreement (e.g. the two examples above) between instructors, we labeled these student answers as having an "unclear" explanation as to the misconception. We view this initial disagreement as an indication that the associated misconception is inherently more challenging to diagnose.

LLM Prompt & Assessment

We crafted a prompt template which clearly articulates the problem given to the students and the policy standards, then asks for an explanation of the student's solution. We incorporated current best practices to include chain-of-thought (CoT) reasoning, see Figure 1. Highlighted text in the figure denotes text that was replaced by the policy solutions at the bottom of the figure or the student's specific answer. The prompt shown represents the final result of multiple iterations of refinement.

Please take on the role of an instructor. You have just given your students a quiz on uniprocessor CPU scheduling policies where you tested their understanding of a particular scheduling policy.
In all policies, at a given timestep, the following order is observed:
1 - newly arrived processes are incorporated to the queue according to the policy
2 - if the process on cpu at the last time step needs to swap, it is taken off and incorporated to the queue according to the policy
3 - the current time step's process is selected (if necessary) from the queue, according to the policy

The processes, arrival times, and service times of the processes follow:

Process	Arrival	Service
A	0	10
B	2	5
C	2 (slightly after B)	7
D	5	3
E	11	1

For each problem, the students need to determine the order in which processes will be scheduled to use the CPU. They will produce a string of the letters A, B, C, D, and E to represent that which processor is running at each increment of time.
The particular policy for this problem was: {selected_policy_expected_solution}
One of the students responded to this problem with the answer: {student_answer}. First, provide a hypothesis of what they imagined the state of the process queue to be at each time step after arrivals are processed and any context switch. It is best if this hypothesis is in the form of a concise table with columns by time step, a row identifying what process is on the processor, and the process queue. Identify where in this table their solution diverges from the correct solution. Second, use this hypothesis to argue whether the student made a random error or a systematic error. Finally, and most importantly: if they made a systematic error, identify what mistaken belief or misconception would explain that error (e.g., student executes context switch prior to processing arrivals at a given time step). Be very concise in your response.

Selected Policy Expected Solutions:
FIFO: "First-In-First-Out CPU scheduling policy. The correct answer should be AAAAAAAAAABBBBBCCCCCCCDDDE."
RRq2: "Round Robin CPU scheduling policy with a quantum of 2. The correct answer should be AABBCCAABBDDCCAABEDCCAACAA."
STCF: "Shortest time-to-completion policy. The correct answer should be AABBBBBDDDCECCCCCCAAAAAAAA."
MLFQ: "A multi-level feedback queue; that uses 3 queues and a quantum of 2 to the power of i where i is the queue number [i in 0,1,2]. In this MLFQ implementation a process which is preempted always goes to a higher quanta / lower priority queue, unless they are already at queue 2 (quanta 4). The correct answer should be AAABCDBBCCDDEAAAABBCCCCAAA."

Figure 1: Main Zero-shot Prompt Template.

We evaluated the responses of the LLMs on each unique, incorrect student answer. For each response, we captured the type of error identified (systematic or random). We then categorized its explanatory value (see Table 1) and whether the LLM's reasoning aligned with our ground truth. In later sections of the paper, we discuss modifications to the base prompt to investigate 1) clearly defining systematic and random errors and 2) the effects of removal of chain of thought requirements.

Large Language Models

We selected the following LLMs: ChatGPT 4.1 [16], Claude Opus 4 [17], and Gemini 2.5 Flash[2] [18]. These were the most capable public models as of June 2025. We explicitly did not include preview models.

[2]Although the Pro version is more comparable to ChatGPT4.1 and Claude Opus 4, it was prohibitively slow, requiring more than 5 minutes for each API call.

Table 1: Explanatory Value Metric

Score	Standard	Quality
0	Little or no value; may be misleading or incorrect.	Useless
1	Some useful analysis, but explains <50% of student solution.	Marginal
2	Explains a majority of the student's solution but is incomplete.	Good
3	Explains all or the vast majority of the student's solution.	Great

Results

In this section, we present the results of our exploration. We begin with the results of our manual analysis of the student answers. Next, we evaluate the LLMs' analysis.

Manual Analysis of Student Answers

We analyzed the answers made by 89 students to the a four-part exam question on CPU scheduling given across two offerings of the course. Table 2 summarizes this analysis for each policy. We include the number of correct answers as well as a detailed breakdown of the incorrect ones. For the incorrect answers, we summarize the total number of such answers, the number of *unique* incorrect answers, the number of *common* incorrect answers (ones made by at least three students), and the number of incomplete answers. Although a threshold of three answers may appear low for defining a misconception as "common," the same incorrect answer provided independently by three students more likely reflects an underlying misconception rather than an isolated or careless error. We omit the incomplete answers (1 RRq2, 6 MLFQ answers) from the remainder of our analysis. In total, we analyzed more than 60 unique, incorrect answers.

			Incorrect Breakdown		
Policy	Correct	Incorrect	Unique	Common	Incomplete
FIFO	86	3	2	0	0
STCF	62	27	11	2	0
RRq2	44	45	20	4	1
MLFQ	44	45	31	2	6
Total	236	120	64	8	6

Table 2: CPU Scheduling Error Analysis

As is evident in Table 2, a higher number of students responded correctly to the simpler FIFO and STCF scheduling policies whereas a higher number of students responded incorrectly to the more difficult RRq2 and MLFQ policies. Further, the more difficult policies have more unique, incorrect answers, perhaps an indication that students have many different misconceptions or make careless or inconsistent errors.

We identified 8 common errors, two each for STCF and MLFQ and four for RRq2 (Table 3). Multiple students making the same mistake likely indicates a shared misconception, which makes the underlying misconception important to identify. With our "ground truth" defined, we now turn to evaluating the performance of GenAI in explaining the students' misconceptions.

Table 3: Incorrect answers made by at least three students (aka common errors)

Policy	Answer	Count
STCF	AABBBDDDBBCECCCCCCAAAAAAAA	12
	AAAAAAAAAADDDEBBBBBCCCCCCC	6
RRq2	AABBCCDDAABBCCDEAABCCAACAA	13
	AAAABBCCAADDBBCCAAEDBCCAAC	4
	AABCABCDABCDABECDABCACACAA	4
	AABBCCDDAABBECCDAABCCAACAA	3
MLFQ	AABBCCDDAABBECCDAABCCAACAA	4
	AABBCCDDAAAAEBBBCCCCDAAAAC	3

Explanatory Power

Our next analysis reveals the agreement between the instructor assessments of student errors and the responses of the three LLMs across the four policies. After reviewing the LLMs explanation of the student misconception for each unique incorrect answer, we assessed whether the LLM response: agreed with the instructor-assessed ground truth; offered a plausible alternative; or was clearly implausible or incorrect. These results are shown in Figure 2.

Notice that as the complexity of the scheduling policy increases, the number of incorrect responses increases; in other words, the variability increases. In addition, the LLM performance drops off sharply with increasing complexity. MLFQ, the most complex of the four policies, shows poor performance, especially as compared to FIFO, the simplest of the four policies.

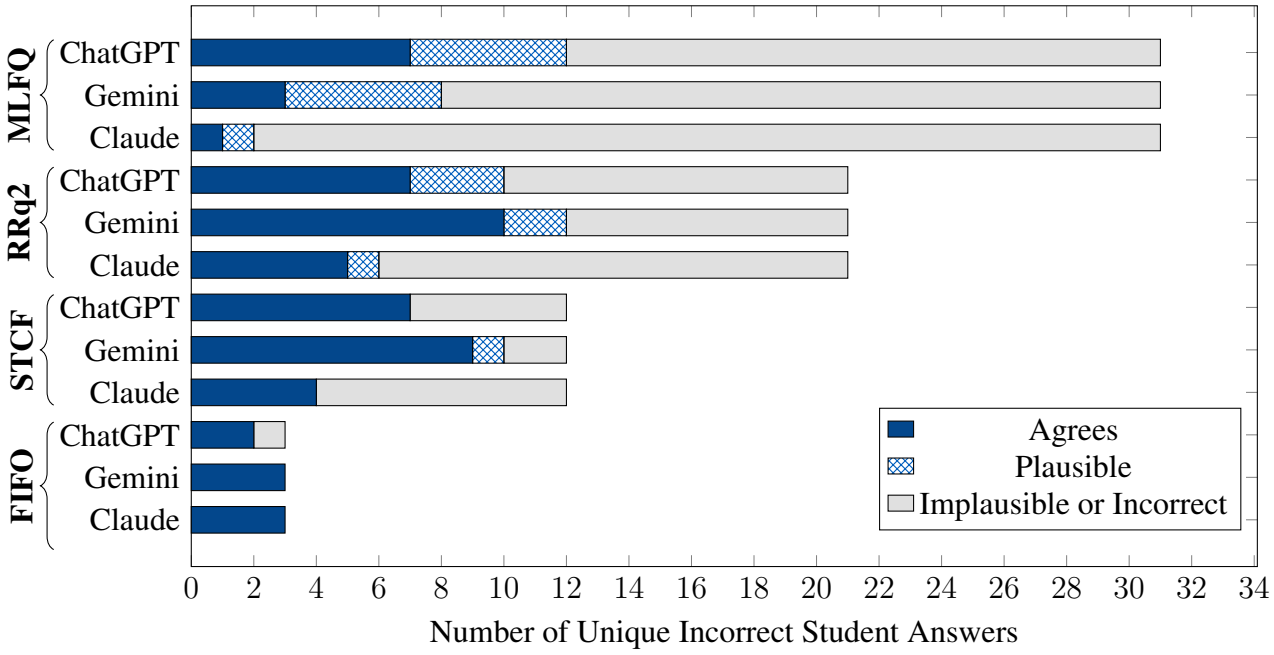

Figure 2: Agreement and Plausibility of LLM Responses

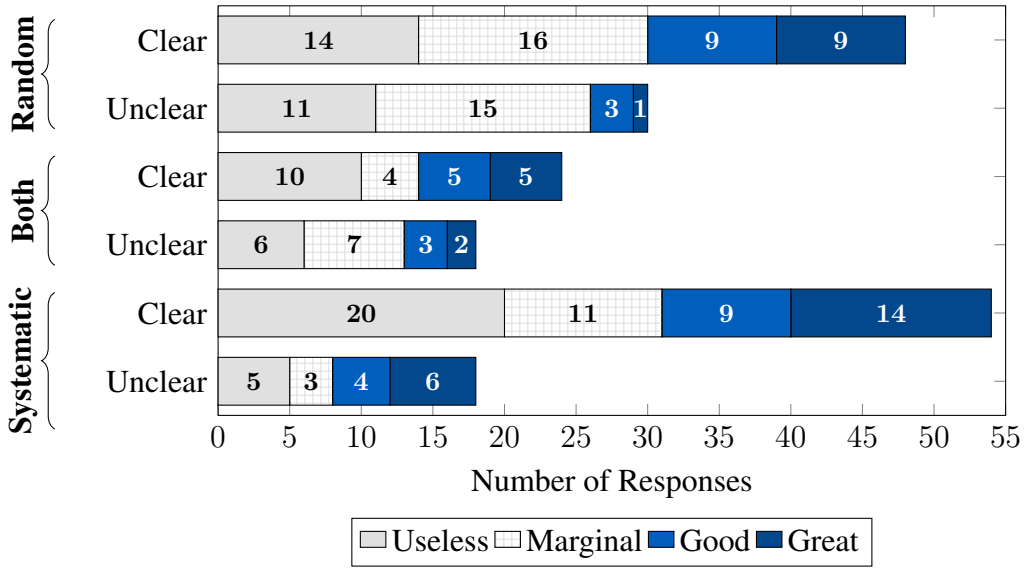

Figure 3: Response quality by error type and initial agreement of manual review. Here "Unclear" populations refer to student solutions where the two manual reviewers initially disagreed on the principle student error.

We next explore how the quality of responses corresponds to the type of error. In Figure 3, six quality of LLM response distributions are plotted, binned by initial agreement of student error (on manual review) and instructor assessment of type of error. If the instructors initially agreed on their manual review of the student error the cases are "clear" ("unclear" if they initially disagreed). The error types are either systematic, random, or both, where "both" indicates there is a systematic error and at least one random mistake.

We note that across all examples, LLM output had a Useless or Marginal rating 63.5% of the time. If we remove Claude from the analysis (because it did uniformly poorly on MLFQ, the largest component of the data set), Useless and Marginal ratings drop to 50%. If we further exclude cases with only random error, we find ChatGPT and Gemini achieve a Good or Great score 60% of the time. This suggests to us that when there is a legitimate misconception to find, current generation LLMs can offer valuable insights, but are overall unreliable especially when confronted with careless errors.

Interestingly, in the 12 cases of systematic error that were "unclear" (ChatGPT and Gemini only), 6 responses were great and 4 were good (84% positive); when compared to the 36 "clear" cases (ChatGPT and Gemini only) there were 12 great and 7 good responses (53% positive). Although there are many qualifiers, we find this to be low-confidence evidence that LLMs may be particularly good in situations when instructors have initial difficulty in diagnosing a student's systematic error.

Type of Error Determination Comparison

A natural question posed by the trend that systematic errors tend to yield better results is "Can an LLM reliably identify the presence of systematic errors in the first place?" In our initial prompt (Figure 1), we did not give a rigorous definition for systematic and random. This resulted in a

<...same as Zero-Shot Main Prompt above here...>

The particular policy for this problem was: {selected_policy_expected_solution}

One of the students responded to this problem with the answer: {student_answer}. Identify whether the student made a random mistake or a systematic error. When making this determination, do not confuse a random error that then cascades into more errors throughout the solution as systematic. Here, systematic means they consistently make the same underlying mistake that leads to their incorrect solution. For example, consider the following two solutions to a FIFO policy:

AABCABCDABCDEABCDABCACACAA - This one is clearly systematic - the student's solution is completely explained if we hypothesize they mistakenly used a round robin scheduling (quanta 1). The idea that they made a random mistake at one location does not explain the recurring pattern of A->B->C, and making many random mistakes to result in a "correct" RRq1 is extremely unlikely.

AAAAAAAAABBBBBCCCCCCCDDDE - This one is very likely just a small random error. There is one fewer A than required, which suggests the student just lost track of the total process time by one for A. If we assume that single error, the rest of the result makes sense and "is correct", even though many other time steps are incorrect in a direct comparison with the correct solution. Solutions like this should NOT be considered systematic simply because the early mistake results in more mistakes later on in the solution. Systematic errors must be a consistent application of a mistaken belief on the scheduling process overall.

What type of error did this student make? Make a short justification for what explains their solution, then say either: Systematic, Random, or Both. Limit your answer to 3 sentences.

Figure 4: One-shot Systematic or Random Prompt

uniformly systematic determination from the LLMs (with one exception), with the LLM's rationale typically being "after the student made a mistake, it cascaded to cause more errors downstream in the solution". For example, for one incorrect answer, ChatGPT gave an explanation that *clearly* showed it understood that the student made a simple mistake, but then concluded that the student had made a systematic error because of the downstream issues. To determine if the LLMs could make a better assessment of type of error with a better explanation, we generated another prompt that has the same initial setup as our original prompt but finishes with a focused question on type of error, with definitions and examples. This prompt is shown in Figure 4.

Using this new data, we next compared the LLMs' determinations to the instructors' "ground truth". For example, if an LLM determined the error to be systematic and the instructor determined the error was random, then that LLM would be marked as incorrect. If the LLM determined an error was systematic and random, but the instructors considered it solely systematic or random, the LLM would be marked as partial. The results of this analysis can be seen in Figure 5. When a student had systematic errors, Claude performed the best, a sharp contrast from our previous experience. When a student made random mistakes, however, performance dropped on average. Notably, Gemini is the only LLM that ever determined that a student's answer had both systematic and random mistakes. We excluded the 9 student answers that the instructors rated as both systematic and random from the chart (Only 1 of the 27 LLM responses from these student answers correctly determined "both" when we did as well, with that one correct assessment from Gemini).

Frequency of Student Errors

We expected that the LLMs would perform better when many students made a particular error because such mistakes are likely not random and may represent more obvious or common

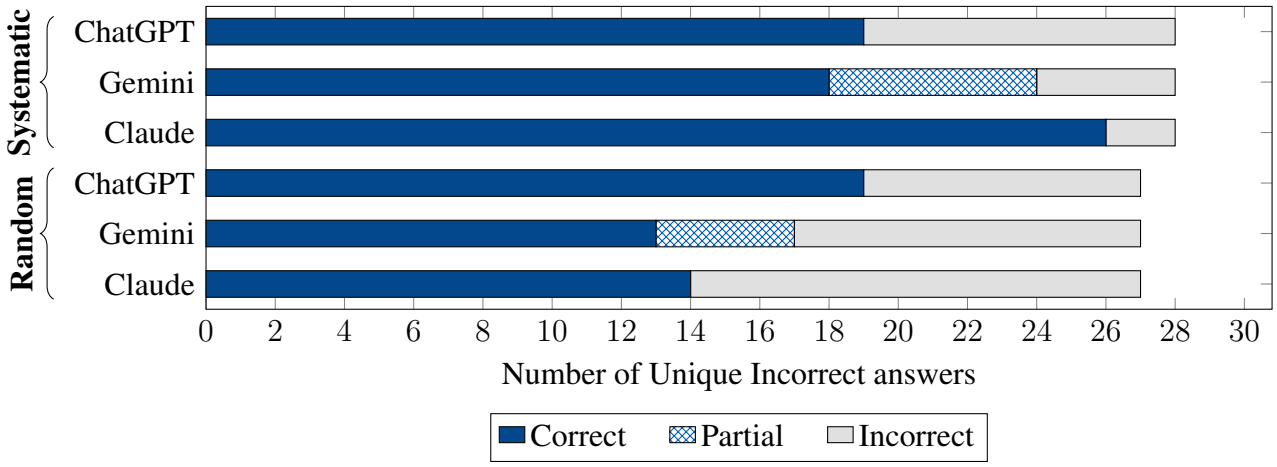

Figure 5: LLM Agreement with Instructors based on Error Type

procedural errors. We did an analysis looking at the common errors shown in Table 3 and any relationship between frequency of the student answer and LLM performance. However, LLMs did not perform better or worse at diagnosing mistakes made by multiple students as compared to singleton errors.

Anecdotal Observations

Beyond the numbers, we also observed some interesting behaviors and utterances from the LLMs. We collect them here for convenience, though some were discussed in more detail earlier.

- ChatGPT misunderstood the quanta length by queue in the MLFQ policy. Similarly, Claude stated that a student ran a process for 11 time units when they had not. LLMs continue to struggle with performing simple arithmetic operations reliably.
- Most of the CoT "state tables" generated by the LLMs to show the hypothesized student reasoning were full of errors. LLMs remain challenged by evaluating logical procedures in a consistent manner.
- ChatGPT asserted that, because a random error then propagated through the remaining answer, the error was systematic.
- ChatGPT repeatedly drew an "always" conclusion from a sample size of one.
- Claude produced numerous, factually incorrect responses.
- Claude correctly identified the one error a student made, but then gave an entirely inaccurate explanation.
- ChatGPT correctly identified that a student might have used RR where arrivals were prioritized, but then gave a summary inconsistent with that observation.

Although it is certainly plausible that additional prompt engineering could improve the performance, we believe our data is representative of what instructors can expect from current generation models in similar tasks.

Reflection

We were encouraged by the LLMs' performance on some tasks but concerned by their inconsistency, particularly on complex policies and random, bookkeeping-style errors. Their feedback largely aligned with Hattie and Timperley's first and second levels of feedback–identifying discrepancies and procedural faults–but never reached the third or fourth levels [19].

LLMs performed best on systematic errors and on simpler policies, but surprisingly, did not improve when analyzing more frequently occurring student mistakes. The disparity in LLM performance between student answers with random versus systematic error may reflect the behaviors and tendency of LLMs to impose coherence on any input. This turns out to be beneficial if the assumption that the student answer demonstrates a mistaken belief is true - but it is misleading when the student's answer contains a random error that then causes the LLM to fabricate an explanation.

We also further explored the role of chain-of-thought (CoT) prompting, which many current best practices recommend. We originally assumed that prompting the LLM to first generate a hypothesis of the student work (CPU state and queues) would increase its attention to the details of the student answer and therefore increase the accuracy and value of the response. However, we found that most of the state tables it generated were full of errors. This domain requires precise logic (and a small amount of simple arithmetic). The demand to infer queue and CPU state proved too challenging for current LLM capabilities, often introducing compounding errors. Once LLM capabilities can better infer such state, we expect CoT techniques provide significant value for this task. We investigated removing the CoT prompt, which reduced detail-driven errors but provided responses that were both more vague and less diagnostic, fixating on the surface-level errors rather than the underlying misconceptions.

Limitations & Future Work

One limitation of this work is we consider one type of exam question focused on a single topic within one course. An area of future work could be to expand the analysis to a larger data set of CPU scheduling questions/answers. Another area of future work would be to extend to other topics within an Operating Systems course as well as other courses within the Computer Science discipline.

We ran into some technical challenges in our use of LLMs. Specifically, we ran into problems with the Gemini API. Twelve of the original 31 unique incorrect answers had to be re-run, seven required 2 re-runs, and two of them required 3 re-runs each to get a complete response. Sometimes the LLM seemed to time out. Sometimes the process-state table it created only contained hyphens. Sometimes it stopped its analysis in the middle of a thought.

Conclusions

The impressive capabilities of GenAI may lead to the assumption that it will be able to determine the cause of student mistakes. Our experiences, however, suggest that GenAI is not yet able to do

this task reliably—certainly not for complex questions. Our findings reinforce those of Balse et al. in their conclusion that "models like GPT-3 currently cannot be 'directly' used to provide feedback to students for programming assessments"[11]. Educators may find value in using LLMs as a first-pass diagnostic tool, especially when the cause of a student's error is unclear, but are cautioned to review the LLMs response for accuracy before giving the feedback to a student. Our findings also suggest that chain-of-thought prompts may be fragile in logic-heavy domains; educators are encouraged to prompt with and without chain-of-thought to see what works best in their particular situation. Our experiences also suggest the need for a broader exploration across courses and across multiple topics within each course to understand the existing boundaries in the area of identifying student misconceptions, and for broader exploration of the prompts used in such queries.

Acknowledgments

During the revision process, we used ChatGPT to provide a review of the paper and to assist in revising our previously written text. ChatGPT also generated the Tikz code to create our figures; all such generated figures were reviewed for accuracy against our data.

The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Army, Department of Defense, or the U.S. Government.

References

[1] S. Z. Bennett-Manke, M. Zhang, and M. R. Ebling. *GraySim*. Accessed: 2026-1/2. 2026. URL: https://github.com/usma-eecs/GraySim.

[2] A. P. Cavalcanti et al. "Automatic feedback in online learning environments: A systematic literature review". In: *Computers and Education: Artificial Intelligence* 2 (2021), p. 100027. ISSN: 2666-920X. DOI: https://doi.org/10.1016/j.caeai.2021.100027. URL: https://www.sciencedirect.com/science/article/pii/S2666920X21000217.

[3] A. Ross and J. Andreas. *Toward In-Context Teaching: Adapting Examples to Students' Misconceptions*. 2024. arXiv: 2405.04495 [cs.CL]. URL: https://arxiv.org/abs/2405.04495.

[4] M. Q. Feldman et al. "Automatic Diagnosis of Students' Misconceptions in K-8 Mathematics". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173838. URL: https://doi.org/10.1145/3173574.3173838.

[5] J. Camacho-Collados and M. T. Pilehvar. "From word to sense embeddings: A survey on vector representations of meaning". In: *Journal of Artificial Intelligence Research* 63 (2018), pp. 743–788.

[6] Y. Shi et al. “Toward Semi-Automatic Misconception Discovery Using Code Embeddings”. In: *LAK21: 11th International Learning Analytics and Knowledge Conference*. LAK21. Irvine, CA, USA: Association for Computing Machinery, 2021, pp. 606–612. ISBN: 9781450389358. DOI: 10.1145/3448139.3448205. URL: https://doi.org/10.1145/3448139.3448205.

[7] M. U. Demirezen, O. Yilmaz, and E. Ince. “New models developed for detection of misconceptions in physics with artificial intelligence”. In: *Neural Computing and Applications* 35.12 (2023), pp. 9225–9251.

[8] S. P. Neshaei et al. “Leveraging Learner Errors in Digital Argumentation Learning: How ALure Helps Students Learn from their Mistakes and Write Better Arguments”. In: *Proc. ACM Hum.-Comput. Interact.* 9.2 (May 2025). DOI: 10.1145/3711023. URL: https://doi.org/10.1145/3711023.

[9] E. Verhelst and T. Belpaeme. “Large Language Models Cover for Speech Recognition Mistakes: Evaluating Conversational AI for Second Language Learners”. In: *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. HRI ’25. Melbourne, Australia: IEEE Press, 2025, pp. 1705–1709.

[10] H. Nguyen, N. Stott, and V. Allan. “Comparing Feedback from Large Language Models and Instructors: Teaching Computer Science at Scale”. In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale*. L@S ’24. Atlanta, GA, USA: Association for Computing Machinery, 2024, pp. 335–339. ISBN: 9798400706332. DOI: 10.1145/3657604.3664660. URL: https://doi.org/10.1145/3657604.3664660.

[11] R. Balse et al. “Investigating the Potential of GPT-3 in Providing Feedback for Programming Assessments”. In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2023. Turku, Finland: Association for Computing Machinery, 2023, pp. 292–298. ISBN: 9798400701382. DOI: 10.1145/3587102.3588852. URL: https://doi.org/10.1145/3587102.3588852.

[12] M. Mueller, C. List, and M. Kipp. “The Power of Context: An LLM-based Programming Tutor with Focused and Proactive Feedback”. In: *Proceedings of the ECSEE 2025: European Conference on Software Engineering Education*. ECSEE 2025. seeon, Germany: Association for Computing Machinery, 2025. URL: https://doi.org/10.1145/3723010.3723034.

[13] M. Liu and F. M’Hiri. “Beyond Traditional Teaching: Large Language Models as Simulated Teaching Assistants in Computer Science”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 743–749. ISBN: 9798400704239. DOI: 10.1145/3626252.3630789. URL: https://doi.org/10.1145/3626252.3630789.

[14] Z. Jiang et al. *LLMs can Find Mathematical Reasoning Mistakes by Pedagogical Chain-of-Thought*. 2025. arXiv: 2405.06705 [cs.CL]. URL: https://arxiv.org/abs/2405.06705.

[15] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. 1.10. Madison, WI, USA: Arpaci-Dusseau Books, Nov. 2023.

[16] OpenAI. *ChatGPT 4.1*. https://chat.openai.com/. Accessed: May-June 2025. 2025.
[17] Anthropic PBC. *Claude Opus 4*. https://www.anthropic.com/claude/. Accessed: May-June 2025. 2025.
[18] Google. *Gemini 2.5 Flash*. https://gemini.google.com. Accessed: May-June 2025. 2025.
[19] J. Hattie and H. Timperley. "The Power of Feedback". In: *Review of Educational Research* 77.1 (2007), pp. 81–112. DOI: 10.3102/003465430298487. URL: https://doi.org/10.3102/003465430298487.