

Are AI Chatbots Proficient at Tracing CS1 code?

Anonymous Authors
University X, University Y
Some city , Some country

ABSTRACT

Learning to write code is linked to the development of code comprehension that captures the low level behaviour of each command, abstracted into a notional machine. CS1 instructors have developed specific tasks, such as Parson Puzzles and reverse tracing, to develop student's code comprehension skills. The emergence of Artificial Intelligence (AI) and Large Language Models (LLMs) have proved to be disruptive technology for computer science education, in both positive and negative ways. The ability to explain and provide code snippets and to answer exam question has been evaluated by multiple studies. Less is known about the tools' ability to perform code tracing and debugging, two key aspects in which students need individual support.

This study aims to address this gap by testing ChatGPT, Copilot and Google Bard with a range of tracing and debugging tasks extracted from the literature. Overall, the study shows tools were successful both at solving code tracing problem at CS1 level and providing descriptive answers that could help students learning to trace. On the other hand, they regularly failed to identify non-trivial bugs, hence they are less suitable to support students when debugging their code.

KEYWORDS

computer science education, cs1, AI code tracing, debugging

ACM Reference Format:

Anonymous Authors . 2024. Are AI Chatbots Proficient at Tracing CS1 code?. In *Proceedings of 29th annual ACM conference on Innovation and Technology in Computer Science Education (ITiCSE 2024)*. ACM, New York, NY, USA, 7 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Students need to dedicate time to learn algorithmic thinking, program comprehension, debugging, and communication skills in order to become proficient computing experts [21, 28]. Developing the ability to understand and explain code is essential for CS students to be competent programmers. It significantly improves students' abilities to write meaningful and efficient code, as they understand how the program is executed at low level. Additionally, code comprehension is also a prerequisite for higher-level code writing skills [18, 24] that are essential for developers, both to explain existing code to new team members or in code reviews, or to explain code

to themselves when they commence working with a new codebase. [17].

However, code comprehension is difficult to master and novices often struggle to develop this skill. Students' struggles are further exacerbated by limited attention and room given for explicit support in CS courses [21]. In addition to programming tasks, CS educators have proposed Parsons Puzzles [9] and a range of tracing exercises to facilitate code comprehension. Tracing helps to gradually develop a mental image of how the computer works - the so called notional machine. Experienced programmers *trace* code in their heads as they write it, and may trace a new algorithm on paper to check for issues. The same behaviour has been observed in students, facilitating the detection of algorithmic issues on students' initial implementations [13].

With the advancement of Large Language Models (LLM) and other Artificial Intelligence (AI) tools, many students have resorted to these tools to help them complete their assigned coursework. The dependence on AI chatbots by students to complete their coursework necessitates research into how these tools can be used effectively. Educators need to make sure the tools support the learning objectives of the coursework instead of providing solutions alone. Thus, we should explore suitable ways to incorporate these chatbots into current CS courses to enhance student learning and improve students' learning outcomes.

This study aims to assess the ability of AI chatbots to aid CS1 students to complete code tracing exercises. In particular, we wanted to investigate the following research questions:

- RQ1. Are AI chatbots able to produce correct outputs for common CS1 tracing tasks?
- RQ2. For correct answers, are the descriptions generated by chatbots detailed enough for students to learn to trace?

The results of this assessment will inform educators on how to adapt the relevant pedagogical approaches in CS1 courses. Effective integration of emergent AI tools will enhance the learning experience for students, leading to increased ability in code comprehension and code development.

2 LITERATURE REVIEW

There have been numerous research efforts that investigated the capabilities of chatbots such as LLMs in completing various types of tasks in CS education. Majority of these research efforts looked into generating code [2, 5, 20?] while others focused on different types of exercises such as computational thinking tasks [4] multiple-choice questions [23, 27], textbook questions [15], exam questions [8] and textual reports [5]. Here we mainly focus on studies that explored the LLM's abilities to provide or critique code explanations in CS tasks.

One such study explored how accurately the GPT-3 model can check, critique and provide suggestions to 1211 student submissions for 7 questions asked in a programming exam. The model could

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ITiCSE 2024, July 08–10, 2024, Milan, Italy

© 2024 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/24/06...\$15.00

<https://doi.org/XXXXXXX.XXXXXXX>

check student code submissions with 57% to 79% accuracy, critique code with 41% to 77% accuracy and suggest appropriate changes with 32%-93% accuracy. These results indicate that such model could not be used ‘directly’ to provide feedback for coding assessments, but may support the instructor to provide one-to-one feedback [3].

Jacques compared the ability of students to explain code compared to LLMs. This focus is incredibly important as students must be able to recognise and explain code they read without needing to compile it. The researchers gathered opinions on the results they collected and discovered that students preferred the code explanations generated by AI chatbots rather than by their peers. The studies show the potential effectiveness of AI when students with to revise or learn by themselves, or if they wish to clarify their understanding of code. However an important limitation of the study is that the code explanations were rated by novices [] and more experienced programmers may prefer a more succinct explanation for code than provided by the AI chatbot. Naturally, novice programmers preferred line-by-line explanations of how a code fragment worked.

Jacques identifies an analogy between Math and Computer Science education. She explains how Math education has developed with the introduction of graphics calculators and other mathematics tools that have made certain aspects of the subject easier for students. These tools are integrated into Math education and are useful tools for students to enhance their education. Jacques supports a similar approach for CS education. AI should become part of the pedagogy in CS1 courses and should be embraced to enhance student learning. This approach is tested by Jacques by allowing students to generate code using chatbots and asking them to modify the code to achieve a more efficient outcome. This approach involves using ChatGPT to aid them in writing code that fits their assignment. This assignment shows an example of how AI can be used successfully in CS1 classrooms by supporting the learning process. There is a focus on the learning itself, rather than the outcome that chatbot generates.

Hassan and Zilles researched the use of reverse-tracing in online examinations. In [11], they conclude that regular code tracing exercises can be brute forced using the AI without issue, or by simply running the given code, making them unsuitable for computerized examinations. Meanwhile, the AI inaccurately completed the reverse-tracing exercises, and it was challenging for students to brute force these exercises. However, this is an isolated example of such an exercise, and would require the development of multiple exercises like this that would be usable in examinations to avoid over-reliance of AI chatbots. There is significant effort required by educators to modify these tasks so that they cannot be brute forced with the help of chatbots. This effort is not negligible and would take up valuable time and resources, especially since different modifications are required for every task.

Daun and Brings [7] explored the use of chatbots for student revision and exam preparation in the area of software engineering. They investigated ChatGPT’s ability to help students understand theoretical concepts while revising such as those taught in software engineering. Their results indicate a variance in the quality of the answers provided by ChatGPT, depending on the specificity of the question and the prompt provided. Yet, ChatGPT struggled to produce useful literature for students to access and read, reducing its

effectiveness in academia. Still, ChatGPT proved to be an excellent means of creating a variety of exercises and questions for both educators and students to utilize in their education. Given the varying competency and ability of ChatGPT to aid in these various tasks, it is essential to use AI appropriately and understand its effectiveness in different situations. Daun and Brings identified that AI may not be used effectively in every kind of learning, which must be recognized by students and educators alike.

Reeves et al. [22] focus on the ability of common generative AI such as Codex to tackle Parson Puzzles, as students can focus on the design algorithmic nature of code without worrying about syntax. The study shows a 50 percent success rate for Codex to solve Parson Puzzles, with results expected to get better over time as new versions of Codex are released. The study acknowledges that prompt engineering plays an important role in the quality of the answers that Codex provides, and that the complexity and style of Parson Puzzle can change the quality and accuracy of answer provided by Codex.

Cipriano and Alves [6] focus on LLM’s ability to aid with Object-Oriented Programming (OOP) problems, which is typically taught in CS2 courses, where programmers have little more experience. Their work shows that GPT-3 can solve OOP problems with minor errors in logic and compilation [6]. The solutions provided were not the best from an OOP perspective but were good enough to pass assessment auto-graders. As the study used Portuguese to pose questions and prompts to GPT-3, it also showed that AI tools can produced satisfactory results even when the questions were not posed in English.

There are still limitations in the use of LLMs, such as being able to plagiarize code, and recognizing that AI will continue to improve and produce better results. Therefore, researchers and educators need to support the use of AI in Computer Science education, rather than resorting to a useless ban. LLMs are also designed with different purposes in mind, which must be acknowledged as their suitability for a task depends on their design. The programming languages with which AI is used may also produce differences in the quality of results, as the amount of training data on a particular programming language will vary.

3 METHODOLOGY

The research was conducted to ensure an in-depth look at the code-tracing capabilities of the different AI Chatbots available, to evaluate their correctness as a first step to use them as tutor support for students learning to trace. Once correctness is established, the study could then be extended further by asking students and CS1 educators to rate the quality of the code explanations provided by the AI.

3.1 AI chatbot selection

A preliminary investigation was conducted by assessing available chatbots to satisfy two criteria. Firstly, it should be available with no paywall or cheap entry cost, and easy to find using major search engines. **Accessibility** to the software ensures that most students, educators and researcher are able to use the resource for their own benefit. Secondly, chatbots should show versatility in **language support**, both in their explanations - to match the native language

of the student - and in the programming language provided to trace - ensuring that the AI can help students regardless of the CS1 language of choice.

Previous studies have shown that most chatbots are multilingual, hence our main concern was accessibility. As such, four AI models were selected in July 2023 for investigation : ChatGPT, Microsoft Copilot (previously named BingChat), Google Bard and Readable-Comment Generator. The first three cover general purpose chatbots while the last is focused on generating documentation for a given piece of code.

3.2 Selection of tracing questions

For the testing of the different chatbots, a code tracing problem set was required. The questions were extracted from previous literature that tested students' code tracing skills. We aim for a set of problems that (1) uses programming languages that are commonly taught in CS1 courses, (2) ranges in complexity (i.e. problems consisting of one or two commands to more complex and nested constructs), and (3) covers a range of core programming concepts (eg. loops, conditionals, function scope) and how they fit together.

The test cases are ordered to match the sequence in which the underlying concepts are generally taught in a CS1 course, the topics covered are listed in Table 1

Initially we selected 8 test cases from tracing papers: three questions in Java, four in Python, and a reverse tracing question from [11]. Test 1 to 7 consist of short code snippets and the students are required to specify the (printed) output. The final test case is a reverse-tracing question, using the same code in *test7*. The tools were tasked with determining potential initial inputs for the program to generate the given output.

We then extended the set with 10 additional problems based a variety of formats and topics: two multiple choice questions from [10], three questions of functions that manipulated lists from [12], and five questions on scope for Python functions from [16]. Acquiring the problems from multiple sources allows for the questions to cover core programming concepts while ensuring code diversity as they are chosen by different instructors and/or researchers.

3.2.1 Extended testing with Refute questions. Tracing often supports the process of testing and debugging code. As most chatbots were able to answer tracing questions, we extended our evaluation with a recently proposed testing task in which students are asked to refute the correctness of a given solution as described in [1]. In their own words "a Refute question presents students with code that fails to achieve a stated task. Students must provide a counter-example input for which the code's behaviour is inconsistent with the task.". The questions cover simple tasks such as Ref3 - Check if a given array contains duplicate elements or Ref4 - Count the unique prime pairs between 0 and N (inclusive); they provide a C++ buggy solution that partially solves that task.

3.3 Testing Procedure

In order to test each AI chatbot, a prompt was developed to request a response. Prompts were identified from [1] and [2]. The prompts used in these papers were considered and distilled down into a singular prompt that could be used for each code snippet, with the exception of reverse tracing. The prompt used for reverse tracing is

Table 1: List of sourced code snippets used for tracing

Prog.	Test	Topic	source	answer type
Java	1	Loop	[26]	output
	2	Nested loops	[26]	output
	3	infinite loop	[26]	output
Python	4	Array update	[19]	output
	5	Array comparison	[19]	output
	6	Recursion	[19]	output
	7	Arrays, filter		output
	8	Arrays, filter <i>reverse tracing</i>	[11]	input
C++	9	Array comparison	[10]	MCQ
	10	nested loops, count	[10]	MCQ
Python	11	List:insert	[12]	output
	12	Loop, List:insert	[12]	output
	13	Loop, List:append	[12]	output
	14	Variable scope	[16]	output
	15	Variable scope	[16]	output
	16	Variable scope	[16]	output
	17	Variable scope	[16]	output
	18	Variable scope	[16]	output

Table 2: Preliminary Test (T1-T8) results

Chatbot	Loops [26]			Arrays []			Rev. [11]
	T1	T2	T3	T4	T5	T7	T8
ChatGPT	✓	✓	✓	✓	✓	✓	✓
Copilot	✓	✓	✓	✓	✓	✓	✓
Google Bard	✓	✓	X	✓	✓	✓	✓
Readable	X	✓	X	X	✓	X	X

tailored for the specific problem. Each chatbot was presented with the code snippet and the same prompt. The prompt used was:

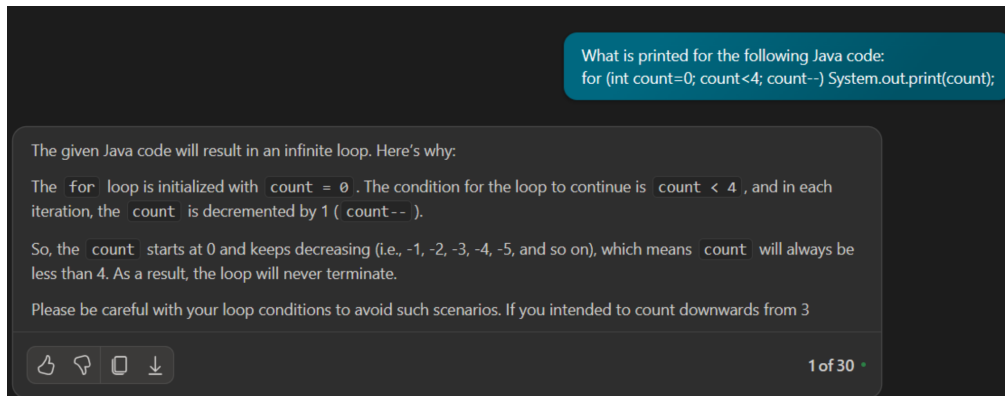
What is printed when the following Java/Python code is run?

This process was repeated for each of pair (test, chatbot) and the responses were recorded. Note we did not extend the prompt to ask for specific explanations or tracing steps, as students will tend to ask for the answer alone. We know from experience that chatGPT will provide not just the answer but also an explanation for it. This is also the case in this experiment, with the chatbot usually describing why/how that particular output would be generated by the code snippet.

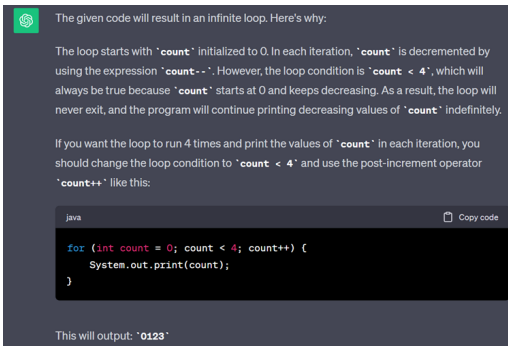
4 RESULTS

4.1 Preliminary testing

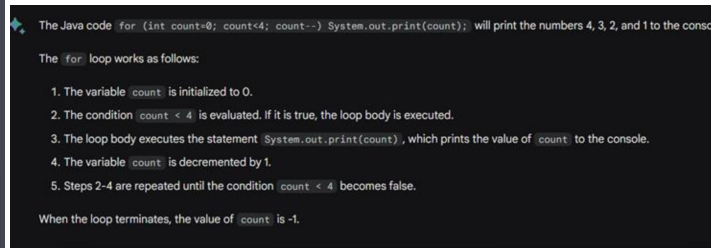
The first set of test was completed with the four chatbots, and their correctness is listed in Table 2. Both ChatGPT and Copilot sporting a 100% accuracy in identifying the right answer to each code tracing problem. Google Bard is slightly less reliable, producing an incorrect answer to *test3*. Finally, Readable is the least reliable of the AI chatbots, passing only two test cases. These results indicate Readable is not well-suited for the purpose of code tracing. Hence, we discard it as a potential tutor for the rest of the study and will no longer be mentioned.



(a) Copilot answer



(b) ChatGPT answer



(c) Google Bard answer

Figure 1: Sample outputs for test case 3.

Table 3: Extended Test (T9-T18) results

Chatbot	C++ [10]	List: insert[12]			Scope in function [16]		
	T9-T10	T11	T12	T13	T14-15	T16	T17-18)
ChatGPT	✓	✓	X	✓	✓	✓	✓
Copilot	✓	✓	✓	✓	✓	X	✓
Google Bard	✓	✓	✓	X	✓	✓	✓

All chatbots provided the correct answer to *test2*, which uses nested loops to print a star pattern [25]. All of them recognised the use of two loops to create a multidimensional array, and then identified that stars were not printed except at the first and last column and row. This test case evidenced their potential capability to understand the use of common structures like nested loops and multidimensional arrays, which is challenging for novice students. Similarly, they were able to interpret user-defined functions with different parameters in the test from [19], including a recursive function in *test6*.

Test 3 is an interesting case as it contains an infinite loop. Fig. 1.(a) shows Copilot answer with the code highlighting the blue question. The answer should say the code keeps printing a list of negative numbers 0, -1, -2 ...etc. ChatGPT produced the correct answer and then suggested a possible fix to eliminate the redundant loop as shown in Fig 1.(b). It is interesting that ChatGPT expands on its answer by suggesting a possible fix, which indicates the ability

of the training ability to develop some debugging capabilities. On the other hand, Google Bard describes the semantics of the loop correctly in steps 1 to 5, but fails to notice the infinite loop as shown in Fig 1.(c). It does not provide an output but the last statement “when the loop terminates the value of *count* is -1” is clearly false.

4.2 Extended testing

Based on the positive results on preliminary testing, we extended the testing to cover C++ tracing question, and to more complex questions in Python: three have methods dealing with list operations [12] and five questions from [16] explore the comprehension of variable scope and side effects; they cover different scenarios in which a local variable and a global variable are assigned the same name.

The results of the test for the three chatbots are shown in Table 3. We can see each chatbot fails a different test. Thus, their performance is reasonable, passing nine out of ten test cases, but not consistent.

Finally, to extend the evaluation of chatbots even further, we asked them to answer five *Refute* questions from [1] as described in section 3.2.1. A summary of the outcomes is provided in Table 4. All chatbots struggled with the refute questions, with Copilot being the most successful by refuting three of the five questions.

With this correction, the code calculates the net value after subtracting each element in the 'losses' list from the initial value of 'net' (which is 100) and then prints the final value of 'net'. Let me calculate it for you:

1. Initial net value: 100
2. After subtracting 1: $100 - 1 = 99$
3. After subtracting 25: $99 - 25 = 74$
4. After subtracting 4: $74 - 4 = 70$
5. After subtracting 9: $70 - 9 = 61$
6. After subtracting 16: $61 - 16 = 45$

Therefore, the output of the code will be:

45

(a) ChatGPT

The Python code you provided is subtracting each value in the list `losses` from the initial `net` value of 100. Here's how it works:

1. Initialize `net` to 100.
2. For each `loss` in `losses` (which are 1, 25, 4, 9, and 16), subtract `loss` from `net`.
3. Print the final value of `net`.

So, if you do the math:

$$100 \text{ (initial net)} - 1 - 25 - 4 - 9 - 16 = 45$$

Therefore, when this code is executed, it will print: 45.

(b) Copilot

Figure 2: Two sample outputs for test case 4

Table 4: Refute Test results

Chatbot	Ref1	Ref2	Ref3	Ref4	Ref5
ChatGPT	✓	X	✓	X	X
Copilot	✓	X	✓	✓	X
Google Bard	X	X	✓	X	X

4.3 Explanations associated with the correct answer

The range of explanations is an interesting area, however space constraints do not allow us to provide at least one sample. We have already observed in Figure 1 that each chatbot use a different variation to explain the simple loop.

Figure 2 shows the answers for *test4* [19] from both ChatGPT and Copilot. Both tools identified that the code looped through an array, and took away the value of the elements in loss from 100. ChatGPT approaches the problem by showing the value of the variable 'net' after each iteration of looping through the arrays shown in Figure 1. In other words, it traces the variable *net*. In comparison, Copilot explained first the initialization of *net* and

then the high level purpose of the loop. Instead of showing each *net* variable update, it abstracts from the description of step2 the list of updates and computed the final value with the math expression.

In this case, the ChatGPT explanation is closer to the tracing explanations given in class, although it does not mention the value of the variable "loss", which is updated in each iteration. However, both answers will help students to build a model of the computation.

5 DISCUSSION

5.1 Research findings

Are AI chatbots able to produce correct outputs for common CS1 tracing tasks? We will answer this question in the positive, as two of the chatbots were able to pass all but one tracing test from a set of 18. However, the failed test belong to extended testing, which is probably beyond the average cs1 tracing task.

The preliminary tests indicate that given code with conditional statements, the three chatbots are able to follow from the initial input and identify when to execute a line of code based on the input value. They were also able to identify a function definition and when the function was called in the code. This is mostly true for extended testing, however each chatbot failed at least one test.

As discussed in section 4.1, *test3*, which contains an infinite loop confused Google Bard, while ChatGPT not only got it correct but it also volunteered a debugging suggestion. It is interesting to note the three chatbots did well at the unusual reverse-tracing question which instead of asking for the output based on the input values, required providing an initial value to the parameter list based on the produced output of a function.

The programming language (C++, Java or Python) used in the code snippets appears to have little or no impact on the chatbot's ability to provide accurate results. In general, these results indicate that ChatGPT and Copilot are the most reliable chatbots for code tracing problems. If we also include the new refute questions, which are challenging for all of them, Copilot appears to be slightly better.

Overall, the results are quite impressive, considering the chatbots are general purpose software. However, as many other studies have revealed, we cannot rely them for accurate results when dealing with complex or unusual scenarios.

For correct answers, are the descriptions generated by chatbots detailed enough for students to learn to trace? Each chatbot provided a detailed explanation with the output as shown in Figures 1 and 2. Although the answers do not trace all the program variables for each line of code, they provide reasonable explanations of the program behaviour. For some tests, eg. *test2* on a nested loop than print starts, both Copilot and ChatGPT provided nearly identical explanations. However, for *test5* ChatGPT provided a very clear explanation of the sequence of two recursive calls made until hitting the base case, while Copilot give only a very short explanation "it calculates the residual of two numbers" and the correct answer. And then for *test6* both provided a high level explanation with no intermediate or tracing values. Overall, ChatGPT tends to give more tracing hint, as illustrated in Figure 2, but not consistently, while Copilot tends to give more abstract answers. In the case of reverse tracing, *test8*, this high-level approach works better.

If students use a wide range of questions, and that the generated answers by chatbots are correct, the chatbots can potentially help learners to gradually develop a correct model of a notional machine.

Tool correctness is important because students will not question the validity of the answers. It is difficult to judge which one provides a better or clearer explanation from the perspective of a novice programmer. The variety of explanations may help students that resort to multiple tools.

5.2 Limitations

One main challenge when evaluating any AI tool is the rapid evolution of Large Language Models (LLMs). Current results may significantly differ from future outcomes, requiring researchers to stay updated in response to technological advancements. Thus, we will expect the tool to improve and pass more tests in the near future.

Additionally, the number of tracing exercises is extensive, hence our problem set may not have cover all possible combinations of code. However, it provides interesting insights on AI capabilities from the tracing perspective, with is a positive contribution to the current considerations about how CS education can be enhanced by incorporating AI chatbots.

6 CONCLUSIONS AND FUTURE WORK

This study has explored the ability of generative AI tools on solving tracing tasks and supporting students to develop both code comprehension and tracing skills. Similar to other studies, we have found that the selected chatbots coped well with a range of tracing questions. The results show that ChatGPT, Copilot, Google Bard are reliable tools to help students with code-tracing problems, and can provide explanations for specific questions. These tools can solve problems relating to a variety of key programming concepts, and can also perform reverse tracing, if needed.

The style in which these explanations are offered can differ from chatbot to chatbot and although it does not trace code line by line, they provide clear explanations key insights into program execution. The analysis of a range of tracing tasks allow educators too understand how these AI chatbots can be used effectively to enhance student learning to program. The applications of these tools in code tracing tasks require educators to consider the viability of code tracing questions and whether they are suitable forms of assessment in a world where students can simply use an AI chatbot to complete the task.

We have also explored the ability of chatbots to detect buggy code, and the results for five refute questions are mixed, may be due to the novelty of the task. We should explore in future work the combined use of tracing, testing and debugging tasks to challenge both students and chatbots to improve their code comprehension and programming skills.

Enhancing CS education by incorporating AI chatbots could make instruction more relevant and tailored to student needs. Additionally, it helps to move from basic recall to deeper comprehension, allowing educators to focus on aspects of students' learning, in a world surrounded by AI.

REFERENCES

- [1] Nimisha Agarwal, Viraj Kumar, Arun Raman, and Amey Karkare. 2023. A Bug's New Life: Creating Refute Questions from Filtered CS1 Student Code Snapshots. In *Proceedings of the ACM Conference on Global Computing Education Vol 1* (<conf-loc>, <city>Hyderabad</city>, <country>India</country>, </conf-loc>) (CompEd 2023). Association for Computing Machinery, New York, NY, USA, 7–14. <https://doi.org/10.1145/3576882.3617916>
- [2] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. 2021. Program Synthesis with Large Language Models. *CoRR* abs/2108.07732 (2021). arXiv:2108.07732 <https://arxiv.org/abs/2108.07732>
- [3] Rishabh Balse, Bharath Valaboju, Shreya Singhal, Jayakrishnan Madathil Warriem, and Prajish Prasad. 2023. 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*. 292–298.
- [4] Carlo Bellettini, Michael Lodi, Violetta Lonati, Mattia Monga, and Anna Morpurgo. 2023. Davinci Goes to Bebras: A Study on the Problem Solving Ability of GPT-3. In *Proceedings of the 15th International Conference on Computer Supported Education, CSEDU 2023, Volume 2, Prague, Czech Republic, April 21-23, 2023*, Jelena Jovanovic, Irene-Angelica Chounta, James Uhomoibhi, and Bruce M. McLaren (Eds.). SCITEPRESS, 59–69. <https://doi.org/10.5220/0012007500003470>
- [5] Robert W. Brennan and Jonathan Lesage. 2023. Exploring the Implications of OpenAI Codex on Education for Industry 4.0. In *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*, Theodor Borangiu, Damien Trentesaux, and Paulo Leitão (Eds.). Springer International Publishing, Cham, 254–266.
- [6] Bruno Pereira Cipriano and Pedro Alves. 2023. GPT-3 vs Object Oriented Programming Assignments: An Experience Report. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 61–67. <https://doi.org/10.1145/3587102.3588814>
- [7] Marian Daun and Jennifer Brings. 2023. How ChatGPT Will Change Software Engineering Education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 110–116. <https://doi.org/10.1145/3587102.3588815>
- [8] Felix Dobsław and Peter Bergh. 2023. Experiences with Remote Examination Formats in Light of GPT-4. In *Proceedings of the 5th European Conference on Software Engineering Education* (Seeon/Bavaria, Germany) (ECSEE '23). Association for Computing Machinery, New York, NY, USA, 220–225. <https://doi.org/10.1145/3593663.3593695>
- [9] 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. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (<conf-loc>, <city>Dublin</city>, <country>Ireland</country>, </conf-loc>) (ITiCSE-WGR '22). Association for Computing Machinery, New York, NY, USA, 191–234. <https://doi.org/10.1145/3571785.3574127>
- [10] Sue Fitzgerald, Beth Simon, and Lynda Thomas. 2005. Strategies That Students Use to Trace Code: An Analysis Based in Grounded Theory. In *Proceedings of the First International Workshop on Computing Education Research* (Seattle, WA, USA) (ICER '05). Association for Computing Machinery, New York, NY, USA, 69–80. <https://doi.org/10.1145/1089786.1089793>
- [11] Mohammed Hassan and Craig Zilles. 2021. Exploring 'reverse-Tracing' Questions as a Means of Assessing the Tracing Skill on Computer-Based CS 1 Exams. In *Proceedings of the 17th ACM Conference on International Computing Education Research* (Virtual Event, USA) (ICER 2021). Association for Computing Machinery, New York, NY, USA, 115–126. <https://doi.org/10.1145/3446871.3469765>
- [12] Mohammed Hassan and Craig Zilles. 2022. On Students' Ability to Resolve Their Own Tracing Errors through Code Execution. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1* (Providence, RI, USA) (SIGCSE 2022). Association for Computing Machinery, New York, NY, USA, 251–257. <https://doi.org/10.1145/3478431.3499400>
- [13] Cruz Izu, Cheryl Pope, and Amali Weerasinghe. 2019. Up or Down? An Insight into Programmer's Acquisition of Iteration Skills. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) (SIGCSE '19). Association for Computing Machinery, New York, NY, USA, 941–947. <https://doi.org/10.1145/3287324.3287350>
- [14] Lorraine Jacques. 2023. Teaching CS-101 at the Dawn of ChatGPT. *ACM Inroads* 14, 2, 40–46. <https://doi.org/10.1145/3595634>
- [15] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. 2023. Chatgpt and software testing education: Promises & perils. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops* (ICSTW). IEEE, 4130–4137.
- [16] Tobias Kohn and Dennis Komm. 2023. Coping With Scoping: Understanding Scope and Parameters. In *Proceedings of the 2023 Conference on Innovation and*

- Technology in Computer Science Education V. 1* (, Turku, Finland,) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 201–207. <https://doi.org/10.1145/3587102.3588798>
- [17] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (, Turku, Finland,) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 124–130. <https://doi.org/10.1145/3587102.3588785>
- [18] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *SIGCSE Bull.* 41, 3 (jul 2009), 161–165. <https://doi.org/10.1145/1595496.1562930>
- [19] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship between Explaining, Tracing and Writing Skills in Introductory Programming. *SIGCSE Bull.* 41, 3, 161–165. <https://doi.org/10.1145/1595496.1562930>
- [20] Victor-Alexandru Padurean, Georgios Tzannetos, and Adish Singla. 2023. Neural Task Synthesis for Visual Programming. *CoRR* abs/2305.18342 (2023). <https://doi.org/10.48550/ARXIV.2305.18342> arXiv:2305.18342
- [21] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education* (<conf-loc>, <city>Turku</city>, <country>Finland</country>, </conf-loc>) (ITiCSE-WGR '23). Association for Computing Machinery, New York, NY, USA, 108–159. <https://doi.org/10.1145/3623762.3633499>
- [22] Brent Reeves, Sami Sarsa, James Prather, Paul Denny, Brett A. Becker, Arto Hellas, Bailey Kimmel, Garrett Powell, and Juho Leinonen. 2023. Evaluating the Performance of Code Generation Models for Solving Parsons Problems With Small Prompt Variations. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (Turku, Finland) (ITiCSE 2023). Association for Computing Machinery, New York, NY, USA, 299–305. <https://doi.org/10.1145/3587102.3588805>
- [23] Jaromir Savelka, Arav Agarwal, Marshall An, Chris Bogart, and Majd Sakr. 2023. Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1, ICER 2023, Chicago, IL, USA, August 7-11, 2023*, Kathi Fisler, Paul Denny, Diana Franklin, and Margaret Hamilton (Eds.). ACM, 78–92. <https://doi.org/10.1145/3568813.3600142>
- [24] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline L. Whalley. 2008. Going SOLO to assess novice programmers. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education* (Paris, France) (ITiCSE '08). Association for Computing Machinery, New York, NY, USA, 209–213. <https://doi.org/10.1145/1384271.1384328>
- [25] Grace Tan and Anne Venables. 2010. Wearing the assessment “bRACElet”. *Journal of Information Technology Education: Innovations in Practice* 9. <https://doi.org/10.28945/707>
- [26] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop* (Berkeley, CA, USA) (ICER '09). Association for Computing Machinery, New York, NY, USA, 117–128. <https://doi.org/10.1145/1584322.1584336>
- [27] Tianjia Wang, Daniel Vargas-Diaz, Chris Brown, and Yan Chen. 2023. Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives. In *IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2023, Washington, DC, USA, October 3-6, 2023*. IEEE, 92–102. <https://doi.org/10.1109/VL-HCC57772.2023.00018>
- [28] Michel Wermelinger. 2023. Using GitHub Copilot to Solve Simple Programming Problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (<conf-loc>, <city>Toronto ON</city>, <country>Canada</country>, </conf-loc>) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 172–178. <https://doi.org/10.1145/3545945.3569830>