Developing *Alice*: A Scaffolding Agent for AI-Mediated Computational Thinking

Muhammad ALI[1*], Bixia CHEN[2], Gary K.W. WONG[3]
[1,2,3]The University of Hong Kong, Hong Kong (S.A.R.)
akula@connect.hku.hk, s1134624@alumni.eduhk.hk, wongkwg@hku.hk

Abstract: *Recent advances in generative AI have increasingly highlighted the transformative potential of large language models (LLMs) within educational contexts. Nevertheless, the token-based generation characteristic of LLMs often results in responses that may lack depth, thereby potentially limiting their effectiveness as scaffolding tools. This paper introduces Alice, a scaffolding agent designed to provide unsolicited hints and adaptive support in computational thinking (CT) education. Alice's effectiveness was primarily evaluated through user feedback scores and benchmarked programming tasks, while further empirical research is underway to explore qualitative evidence of scaffolding effectiveness. Alice was initially optimized for both plugged and unplugged learning scenarios using a structured system prompt informed by a hierarchical framework for AI-mediated CT and the oDSP-HF approach. Subsequent fine-tuning with a LoRA-based method reduced perplexity from 9.5 to 6.6 and improved JavaScript-to-block-based code conversion accuracy from 45.8% to 69.5%. User ratings also increased from 64% to 85%. These findings tentatively indicate that careful system prompt design, combined with targeted fine-tuning, may enhance the adaptive support and learner engagement provided by LLM-based scaffolding agents in CT education.*

Keywords: computational thinking, artificial intelligence (AI), scaffolding agent, large language models, fine-tuning

1. Introduction

The rapid advancement of generative AI (GenAI) has arguably garnered significant interest in its potential to enhance personalized learning, instructional design, and assessment (Khosravi et al., 2023). Within this context, a key area of scholarly exploration concerns AI-powered scaffolding agents that dynamically adapt instructional content and provide real-time, personalized hints—often referred to as "unsolicited scaffolds" (Hijón-Neira et al., 2023). However, despite such promise, substantial challenges arise from the inherent limitations of GenAI technologies, particularly large language models (LLMs), which operate primarily through token prediction rather than explicit reasoning (Wei et al., 2022). While LLMs appear to demonstrate strong language generation capabilities, they are also prone to hallucinations and frequently lack contextual awareness. Unlike human writers, LLMs do not autonomously seek clarification on task requirements, a limitation that has led Hicks et al. (2024) to characterize them as "bullshit generators." Although prompt engineering techniques, such as "CoT prompting," have been developed to mitigate these issues (Wei et al., 2022), current evidence suggests that such strategies remain largely insufficient for providing meaningful scaffolding.

Beyond technical limitations, the developmental trajectory of LLMs further complicates their role as scaffolding agents. As these models become increasingly optimized for information retrieval rather than interactive learning (Khosravi et al., 2023), they typically provide direct answers instead of engaging in inquiry-based scaffolding. Effective scaffolding, however, arguably requires assessing prior knowledge, posing counter-questions, and fostering metacognitive engagement—pedagogical affordances that are often absent in current LLM interactions. Chen et al. (2023) emphasize the need for AI-powered scaffolding agents to align with pedagogical affordances, including reflectivity and effortful engagement, thereby necessitating both new human competencies and safeguards against unintended consequences. This context raises the first critical research question (**RQ1**): **How can AI-powered scaffolding agents be effectively developed to align with key pedagogical affordances?** One potential approach involves designing structured “system prompts” to guide LLMs in scaffolding interactions (Zhang et al., 2024). Nevertheless, research on translating prompt engineering into practical educational applications remains limited.

These challenges become even more pronounced at the intersection of AI and computational thinking (CT) education. CT has long been recognized as a crucial problem-solving skillset, with its conceptual roots in Seymour Papert's work (Lodi & Martini, 2021). While Jeannette Wing popularized the term, CT remains an evolving construct shaped by multiple frameworks, which arguably necessitates a reconsideration of Papert's foundational ideas to adapt them to emerging educational needs (Wong et al., 2020). Papert's constructionist vision emphasized hands-on engagement with computational artifacts as a means of fostering problem-solving skills, positioning CT as a mindset for leveraging computational technologies—including AI—to explore and solve problems. Despite early synergies between CT and AI, it may be observed that modern discourse has yet to fully articulate their relationship. Given AI's increasing presence in education, clarifying how it mediates CT learning while preserving human cognitive processes remains a pressing concern.

One area in which GenAI has been explored in CT education is programming-related learning activities. Researchers have integrated LLMs into block-based coding environments to support CT instruction, aligning with Papert's emphasis on active engagement with computational artifacts (Ali et al., 2024). LLMs may assist students in understanding computational concepts and generating text-based code, yet notable challenges persist. For instance, Kong et al. (2024) indicate that LLMs often struggle to accurately describe CT concepts, and students frequently misinterpret AI-generated code. Despite these limitations, LLMs arguably hold potential as cognitive scaffolds in CT education, particularly in programming contexts.

In response to these developments, scholars have called for new CT frameworks that integrate AI to support computational problem-solving (Ali et al., 2024; Wong et al., 2020). To encapsulate this evolving relationship, we propose ‘AI-mediated CT’ as a hierarchical framework that seeks to extend Papert's vision by positioning AI as a cognitive tool that supports—rather than supplants—human cognition. AI-mediated CT reinforces the constructionist principle that learners should remain active agents in their own learning process. Within this framework, AI functions as a cognitive scaffold, offering hints, feedback, and adaptive support to enhance computational problem-solving.

This conceptualization gives rise to the second critical research question (**RQ2**): **How can AI-mediated CT be pragmatically structured as a framework to guide system prompt design for LLMs?** Addressing this question necessitates a logically organized framework that accommodates both plugged (programming-related) and unplugged (non-programming-related) learning scenarios. The following section delineates the conceptual foundations of the research project, thereby establishing the basis for the development of *Alice*—the AI-mediated CT scaffolding agent.

2. Conceptual Groundwork

2.1. Optimizing Directional Stimulus Prompting Through Human Feedback: A Structured Approach

This subsection addresses **RQ1** in part by drawing upon contemporary technical research to propose a structured approach for designing system prompts that enables LLMs to exhibit key pedagogical affordances essential for effective scaffolding. As previously highlighted, such affordances include promoting reflectivity, fostering metacognition, and encouraging effortful engagement in learner interactions (Chen et al., 2023). In this context, system prompts refer to embedded instructions that persist in the background each time an LLM is initialized for a new inquiry, thereby defining the scope of interaction without altering the model's internal parameters (Zhang et al., 2024).

It may be argued that system prompts can be strategically designed to achieve these pedagogical affordances by instructing LLMs to avoid providing direct answers. Instead, these prompts facilitate a controlled conversational process involving hint generation, meaningful questioning, iterative refinement based on user feedback, and collaborative solution derivation. This structured approach, delineated here as *Optimizing Directional Stimulus Prompting Through Human*

Feedback (oDSP-HF), builds upon the "hint generation" concept in Directional Stimulus Prompting (DSP) proposed by Li et al. (2023) for language summarization tasks.

Unlike DSP, which relies on a secondary model, oDSP-HF operates using only a single LLM, integrating direct human interaction to refine hint generation. Upon receiving an inquiry, the LLM generates preliminary hints, which are then iteratively refined through user feedback until a clear problem-solving direction emerges. Once the user is satisfied, the LLM synthesizes the refined hints with the original inquiry to generate the final response. This iterative exchange is posited to foster a collaborative problem-solving dynamic, thereby enhancing both guidance and understanding.

Although oDSP-HF appears to hold potential for a variety of scaffolding tasks beyond AI-mediated CT, its implementation in this context introduces unique complexities. Unlike language summarization tasks, hint generation for AI-mediated CT arguably requires a more nuanced approach to system prompting. The following subsection, therefore, explicates a hierarchical framework designed to guide system prompting with oDSP-HF, thereby ensuring better alignment with the expectations of AI-mediated CT.

2.2. A Hierarchical Framework for AI-Mediated CT

This subsection addresses **RQ2** by expanding on the conceptualization of AI-mediated CT discussed in Section 1. AI-mediated CT frames AI as a scaffolding agent that provides unsolicited hints based on human feedback, augmenting computational reasoning and supporting computational problem-solving. However, as previously noted, CT—let alone *thinking* itself—is not as well-defined as strictly outcome-driven tasks such as language summarization. Thinking is highly dynamic and varies across individuals (Shin, 2019), making it inherently more complex to scaffold effectively.

A pragmatic approach to addressing this challenge is to refine the focus from 'thinking' to 'reasoning.' While thinking encompasses a broad set of cognitive processes, reasoning relies on logic or structured rules to draw inferences (Shin, 2019). By approximating human reasoning, an AI may, in principle, scaffold human thinking to a practical extent. This suggests that thinking can be conceptualized as a hierarchy of reasoning complexity, ranging from general information retrieval to contextualized problem-solving. However, when applied to CT, the absence of a structured framework becomes evident. To the best of our knowledge, no existing framework explicitly enables AI to approximate CT, even hypothetically. While multiple interpretations of CT exist, they vary in alignment and offer holistic perspectives on key computational concepts such as algorithms, decomposition, iteration, abstraction, and debugging. Yet, an epistemological gap remains in understanding how these concepts interrelate with computational reasoning and problem-solving.

To bridge this gap, *three* CT experts in computer science, AI, and education systematically reviewed existing frameworks and identified overlapping dimensions, including only constructs with 100% interrater agreement. This rigorous process underscored the need for a dynamic, hierarchical framework applicable to both plugged and unplugged learning contexts. The resulting framework comprises *four* key dimensions, namely Computational Reasoning, Computational Concepts, Computational Practices, and Computational Constructs, as illustrated in Figure 1.

At the base of this hierarchy is ***Computational Reasoning***, which is argued to be a cornerstone of CT. Computational reasoning relies on formal logic to represent computational problems, address them systematically, and derive conclusions (Paulson, 2018). Researchers assert that LLMs exhibit emergent abilities to approximate human reasoning, particularly computational reasoning, to a noticeable extent (Wei et al., 2022). Therefore, computational reasoning serves as a crucial link between AI and CT, positioning it as the fundamental dimension for integrating the hierarchical framework with the oDSP-HF approach to guiding the system prompting of LLMs for scaffolding AI-mediated CT.

The next level, ***Computational Concepts***, applies computational reasoning to general problem-solving contexts. Concepts such as algorithms, decomposition, iteration, and abstraction become relevant at this stage (Ali et al., 2024). Since different CT frameworks define these concepts in varying ways, there is no fixed set of computational concepts. Following this, the hierarchy progresses to ***Computational Practices***, which involve applying computational concepts to specific problem-solving scenarios. For example, decomposition—the process of breaking down a problem into smaller parts—is a computational concept. However, when applied in a structured problem-solving task, decomposition becomes a computational practice (Ali et al., 2024). As a computational practice in the development of a pathfinding algorithm, 'decomposition' involves breaking the problem into subproblems such as graph representation, node traversal, and cost evaluation. Each subproblem is addressed individually before being integrated into a complete solution.

The final level, ***Computational Constructs***, represents the implementation of computational practices within programming environments. Iteration, for instance, functions as both a computational concept and a computational practice, involving the repeated execution of a process until a condition is met (Ali et al., 2024). However, when implemented in programming, iteration is operationalized through constructs such as FOR loops, WHILE loops, or recursive function calls. In search algorithms, these constructs provide the syntactic and structural mechanisms necessary to execute iteration in code. Thus, 'iteration' transitions from a computational practice to a formalized computational construct within a programming language.

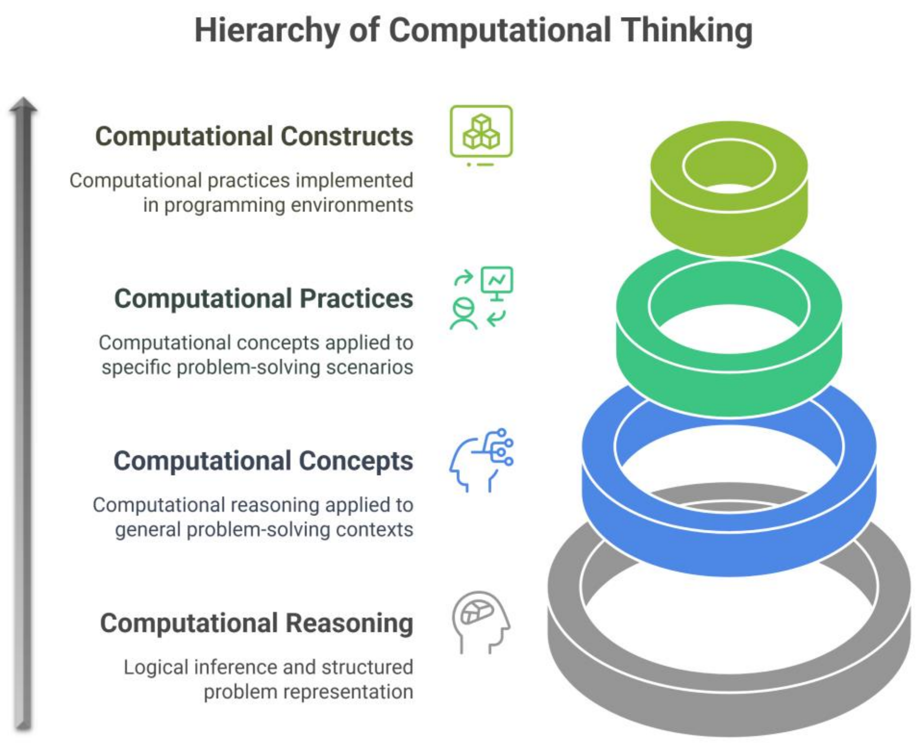
Hierarchy of Computational Thinking
Computational Constructs
Computational practices implemented in programming environments
Computational Practices
Computational concepts applied to specific problem-solving scenarios
Computational Concepts
Computational reasoning applied to general problem-solving contexts
Computational Reasoning
Logical inference and structured problem representation

Figure 1. The hierarchical framework for AI-Mediated CT.

Although not intended as a definitive model, this pragmatic framework (see Figure 1) serves to structure AI-mediated CT for system prompt design and scaffolding. The following section outlines *Alice*'s technical implementation using the oDSP-HF approach, which is informed by the hierarchical framework, as well as the subsequent fine-tuning and performance evaluation.

3. Technical Implementation

Alice, the AI agent, is built upon Llama 3.1 (70B), an open-source LLM developed by Meta. This model was selected for its adaptability, its support for fine-tuning, and its competitive long-context reasoning capabilities, all of which are considered essential for structured system prompting and are understood to align closely with proprietary models such as GPT-4.

For the initial implementation (**RQ1**), the system prompt was designed following the oDSP-HF approach, as guided by the hierarchical framework for AI-mediated CT. The prompt was crafted to specify *Alice*'s persona, incorporate DSP instructions, and include few-shot CoT exemplars to reinforce computational reasoning. To ensure contextually relevant hints for both plugged and unplugged learning scenarios, the system prompt was layered according to the four hierarchical dimensions of AI-mediated CT. *Alice* was deployed using Poe's server deployment feature (https://creator.poe.com/docs/), and hosted on a local server. This approach was intended to provide greater control over deployment, data management, and iterative refinements.

Alice's response accuracy, logical coherence, and adherence to the system prompt were systematically assessed following deployment. Over a period of three months, 64 K–12 trainee teachers enrolled in a CT education course at a prominent university in Hong Kong interacted with *Alice* via Poe. The course encompassed both plugged learning scenarios (e.g., Micro-Bit coding) and unplugged learning scenarios (e.g., LEGO patterns, mind mapping). In accordance with Poe's data privacy policy, only user interactions (prompt-response pairs) were recorded, with all personal identifiers excluded. This process yielded a total of 10,346 interactions, and user ratings were collected through Poe's "thumbs-up/thumbs-down" feedback system.

For the fine-tuning phase, interactions were categorized into four groups: positively rated, negatively rated at the end of conversations, neutrally rated at the end of conversations, and remaining interactions. It is notable that over 84% of negative interactions involved Micro-Bit JavaScript-to-block-based code conversion issues, which appears consistent with prior research indicating that LLMs are not specifically fine-tuned for such tasks (Kong et al., 2024). A dataset of 3,661 prompt-response pairs was curated and manually labeled for fine-tuning, which was conducted using Low-Rank Adaptation (LoRA) via the Unsloth library (https://docs.unsloth.ai/). LoRA, a parameter-efficient fine-tuning method (Hu et al., 2021), is designed to reduce computational overhead while maintaining model performance. The fine-tuning process employed a LoRA rank of 16, a LoRA alpha of 16, and a LoRA dropout of 0.1, targeting key projection modules.

Training was performed on dual RTX 3090 Ti GPUs using the AdamW optimizer, with a batch size of 2, gradient accumulation steps of 10, a learning rate of $1.5e^{-4}$, and 3 epochs, completing in 28 hours. Following fine-tuning, a validation set of 550 samples, including 25% Micro-Bit-specific queries, was used to evaluate performance. Results from *Alice*'s fine-tuning are summarized in Table 1.

Table 1. Alice's performance pre- and post-fine-tuning.

Metric	Pre-Fine-Tuning	Post-Fine-Tuning
Perplexity (validation set)	9.5	6.6
Code conversion accuracy (all, %)	45.8 (±3.1)	69.5 (±2.5)
Code conversion accuracy (unseen, %)	37.2 (±3.4)	64.8 (±2.7)
User thumbs-up rate (%)	64	85

As indicated in Table 1, fine-tuning led to substantial improvements in *Alice*'s performance. The reduction in perplexity suggests enhanced fluency and coherence, while the marked increase in JavaScript-to-block-based code conversion accuracy and user thumbs-up rates arguably indicates more effective scaffolding. These improvements were particularly pronounced in previously unseen problems, highlighting *Alice*'s strengthened generalization capabilities. These results indirectly support the effectiveness of the oDSP-HF approach and the hierarchical framework in guiding AI-mediated CT scaffolding.

4. Conclusion

In evaluating *Alice*'s AI-mediated CT scaffolding performance, it should be emphasized that no direct quantitative metrics were available; rather, user feedback trends served as indirect performance indicators over time. To gain a deeper understanding of these observed improvements, an interpretive inquiry is currently underway to explore *Alice*'s integration within CT teacher education settings. While preliminary findings underscore the potential of AI-powered scaffolding in CT education, they also draw attention to ongoing challenges related to agent adaptability and the contextual specificity required for effective implementation. Nevertheless, our ongoing research continues to focus on refining the oDSP-HF approach to system prompting, with the broader aim of advancing the development of more sophisticated scaffolding agents capable of supporting AI-mediated CT and related educational applications.

References

Ali, M., Wong, G. K.-W., & Ma, M. (2024). K–12 Pre-service Teachers' Perspectives on AI Models and Computational Thinking: The Insights from an Interpretative Research Inquiry. *Proceedings of the 8th APSCE International Conference on Computational Thinking and STEM Education (CTE-STEM 2024)*, *8*(1), 66-71. https://doi.org/10.5281/zenodo.11559685

Chen, B., Zhu, X., & Díaz del Castillo H, F. (2023). Integrating generative AI in knowledge building. *Computers & Education: Artificial Intelligence*, *5*, 100184. https://doi.org/10.1016/j.caeai.2023.100184

Hicks, M. T., Humphries, J., & Slater, J. (2024). ChatGPT is bullshit. *Ethics and Information Technology*, *26*(2), 38. https://doi.org/10.1007/s10676-024-09775-5

Hijón-Neira, R., Connolly, C., Pizarro, C., & Pérez-Marín, D. (2023). Prototype of a Recommendation Model with Artificial Intelligence for Computational Thinking Improvement of Secondary Education Students. *Computers (Basel)*, *12*(6), 113. https://doi.org/10.3390/computers12060113

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *arXiv [cs.CL]*. https://doi.org/10.48550/arXiv.2106.09685

Khosravi, H., Viberg, O., Kovanovic, V., & Ferguson, R. (2023). Generative AI and Learning Analytics. *Journal of Learning Analytics*, *10*(3), 1-6. https://doi.org/10.18608/jla.2023.8333

Kong, S.-C., Sit, E. C. Y., Yang, N. Y., & Yeung, W. K. (2024). One Step Forward towards the Use of Human Language to Instruct ComputerstoWork: A Reflection on an Example of Applying Prompts in Text-based GenerativeAI for Programming. In *Proceedings of the 8th APSCE International Conference on Computational Thinking and STEM Education (CTE-STEM 2024)* (pp. 151-153), Beijing, China.

Li, Z., Peng, B., He, P., Galley, M., Gao, J., & Yan, X. (2023). Guiding Large Language Models via Directional Stimulus Prompting. *arXiv [cs.CL]*. https://doi.org/10.48550/arXiv.2302.11520

Lodi, M., & Martini, S. (2021). Computational Thinking, Between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

Paulson, L. C. (2018). Computational logic: its origins and applications. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*(2210), 20170872-20170872. https://doi.org/10.1098/rspa.2017.0872

Shin, H. S. (2019). Reasoning processes in clinical reasoning: from the perspective of cognitive psychology. *Korean Journal of Medical Education*, *31*(4), 299-308. https://doi.org/10.3946/kjme.2019.140

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022)*, *36*(1), 24824-24837. https://dl.acm.org/doi/10.5555/3600270.3602070

Wong, G. K.-W., Ma, X., Dillenbourg, P., & Huan, J. (2020). Broadening artificial intelligence education in K-12: Where to start? *ACM Inroads*, *11*(1), 20-29. https://doi.org/10.1145/3381884

Zhang, L., Ergen, T., Logeswaran, L., Lee, M., & Jurgens, D. (2024). SPRIG: Improving Large Language Model Performance by System Prompt Optimization. *arXiv [cs.CL]*. https://doi.org/10.48550/arXiv.2410.14826