Hierarchical Multi-Agent Orchestration for Interpretable Educational Dialogue: Emergent Reasoning in LLM-Powered Tutoring Systems

Anonymous Submission

Abstract

Large Language Models (LLMs) have enabled conversational tutoring systems, yet they struggle with maintaining logical consistency, providing specialized domain expertise, and offering interpretable reasoning across multi-turn educational dialogues. We present a hierarchical multi-agent architecture that decomposes educational dialogue into coordinated interactions among 18+ specialized agents organized across four tiers: perception, domain expertise, coordination, and strategic planning. Unlike monolithic approaches where a single model attempts all educational functions simultaneously, our framework treats adaptive tutoring as an emergent property of agent collaboration with explicit consistency constraints and interpretable decision-making. Through deployment on a learning platform serving thousands of users and evaluation on established reasoning benchmarks, we demonstrate that agent orchestration achieves substantial improvements: 15.1% higher accuracy on deductive reasoning tasks (ProofWriter), 66.1% reduction in temporal inconsistencies across dialogue sessions, and 83.9% deeper multi-turn reasoning chains while maintaining interpretability through agent specialization. We provide architectural principles for decomposing complex dialogue tasks, coordination protocols ensuring consistency, and analysis of emergent behaviors including cross-domain collaboration and adaptive scaffolding. Our work contributes both empirical evidence that hierarchical agent orchestration addresses fundamental limitations in educational dialogue systems and theoretical insights into interpretability through architectural decomposition.

1 Introduction

The integration of Large Language Models into educational applications has enabled sophisticated conversational tutoring systems that can explain concepts, answer questions, and provide feedback in natural language (5; 4). However, educational dialogue presents unique challenges that expose fundamental limitations in current LLM-based approaches:

Logical Consistency Across Sessions Unlike casual conversation, educational dialogues require maintaining logical coherence across extended interactions. A tutoring system that teaches "binary search requires sorted arrays" in one turn cannot later claim the opposite without undermining learning. Recent work shows that even state-of-the-art LLMs produce contradictions in 18.4% of multi-turn technical explanations (25; 10). **Domain Expertise Specialization**: Educational content spans diverse technical domains—from algorithm analysis requiring formal complexity reasoning to distributed systems demanding understanding of CAP theorem tradeoffs. Monolithic models trained on next-token prediction lack mechanisms for deep specialization, leading to superficial or incorrect explanations in technical domains (13; 8). **Interpretability and Trust**: Students and educators need to understand *why* a tutoring system provides specific feedback or recommendations. The "black box" nature of monolithic LLMs undermines trust and prevents meaningful pedagogical evaluation (1; 3).

Current approaches attempt to address these limitations through enhanced prompting strategies (31), tool integration (26), or fine-tuning on educational data. However, these methods remain fundamentally reactive, lack explicit consistency enforcement, and provide limited interpretability into reasoning processes.

1.1 Our Approach: Hierarchical Agent Orchestration

We propose treating educational dialogue as a *multi-agent coordination problem* where special-

ized LLM-powered agents collaborate to generate adaptive, consistent, and interpretable learning experiences. Our architecture organizes agents into four tiers:

1. **Perception Agents**: Continuously monitor learner behavior, detecting engagement patterns, error sequences, and affective states

2. **Domain Agents**: Provide specialized expertise in programming, system design, and mathematical reasoning, each delegating to sub-agents for fine-grained analysis

3. **Coordination Agents**: Orchestrate multi-agent responses, enforce consistency constraints, and aggregate domain-specific outputs

4. **Strategic Agents**: Generate long-term learning roadmaps, predict struggle areas, and recommend proactive interventions

This hierarchical decomposition enables *interpretability through architecture*: rather than attempting to explain opaque model internals, we make the reasoning process transparent through explicit agent specialization and coordination protocols.

1.2 Contributions

This paper makes four key contributions:

1. **Novel Architecture for Educational Dialogue**: A formal framework for hierarchical multi-agent orchestration with explicit consistency constraints operating across multiple temporal scales (Section 4)

2. **Comprehensive Evaluation**: Systematic evaluation on established reasoning benchmarks (ProofWriter, MATH, LogiQA) and large-scale deployment serving thousands of learners, demonstrating improvements in accuracy, consistency, and engagement (Section 5)

3. **Analysis of Emergent Behaviors**: Identification and characterization of emergent coordination patterns including cross-domain collaboration, adaptive proof strategies, and collaborative error recovery that arise without explicit programming (Section 7)

4. **Interpretability Through Decomposition**: Demonstration that architectural specialization provides inherent interpretability.

2 Related Work

2.1 Educational Dialogue Systems

Intelligent Tutoring Systems (ITS) have long pursued personalized education through adaptive instruction (28; 2). Recent work leverages neural models for knowledge tracing (22; 9) and dialogue management (20). However, most systems employ monolithic architectures that struggle with domain specialization and consistency maintenance across extended interactions.

2.2 LLM-Based Tutoring

The emergence of LLMs has enabled new tutoring paradigms. Khan Academy's Khanmigo (16) provides conversational support, while various systems explore LLMs for math tutoring (30), code explanation (17), and conceptual learning (15). Recent work by Chudziak and Kostka (6) demonstrates adaptive math tutoring with graph-based knowledge retrieval, while Zhang et al. (34) simulate classroom dynamics with LLM-empowered agents.

However, these systems remain fundamentally reactive and lack formal mechanisms for maintaining logical consistency or providing specialized domain expertise. Our work addresses these gaps through hierarchical agent orchestration.

2.3 Multi-Agent Systems and LLMs

Classical multi-agent research established frameworks for distributed problem solving (32; 11). Recent work has begun exploring LLM-powered agents: Park et al. (21) simulate human behavior through agent coordination; Wu et al. (33) enable multi-agent conversations for code generation; Hong et al. (14) implement software development through role-specialized agents; Qian et al. (23) support collaborative engineering.

While promising, these systems focus on task completion rather than educational dialogue requirements: maintaining pedagogical coherence, enforcing logical consistency, and providing interpretable reasoning chains. Our architecture specifically addresses educational constraints through formal consistency protocols and temporal stratification.

2.4 Explainability in NLP Systems

Recent work explores post-hoc explanation methods (24; 19), attention visualization (7; 29), and chain-of-thought prompting (31). However, these approaches attempt to explain opaque model internals.

We contribute an alternative paradigm: *interpretability through architectural decomposition*. Rather than explaining what a single model does, we make the reasoning process transparent through explicit agent specialization, communication protocols, and coordination traces.

3 Problem Formulation

We formalize educational dialogue as a multi-agent sequential decision problem with explicit consistency constraints.

3.1 Educational Dialogue Task

Let $\mathcal{L}$ represent a learner with evolving knowledge state K_t at time t. An educational dialogue system engages in multi-turn interaction:

$$\mathcal{D} = \{(q_1, r_1), (q_2, r_2), \ldots, (q_T, r_T)\} \quad (1)$$

where q_t is the learner's query and r_t is the system's response at turn t. Unlike general dialogue, educational interactions must satisfy three critical properties:

P1: Temporal Consistency Responses must remain logically consistent with previous turns:

$$\forall t_1 < t_2 : r_{t_1} \wedge r_{t_2} \not\models \bot \quad (2)$$

P2: Domain Accuracy Responses must reflect specialized domain expertise $\mathcal{E}_d$ for domain d:

$$r_t \in d \Rightarrow r_t \models \mathcal{E}_d \quad (3)$$

P3: Adaptive Scaffolding Response complexity should match learner state:

$$complexity(r_t) = f(K_t, q_t) \quad (4)$$

3.2 Multi-Agent Formulation

We decompose the dialogue task across specialized agents $\mathcal{A} = \{A_1, \ldots, A_n\}$. Each agent A_i is characterized by:

- **Domain expertise** D_i: Specialized knowledge (e.g., algorithms, distributed systems)
- **Reasoning type** R_i: Deductive, inductive, or abductive inference
- **Temporal scope** T_i: Operational timescale (milliseconds to months)
- **Consistency constraints** C_i: Local logical rules the agent must satisfy

A coordination protocol Π maps queries to agent orchestrations:

$$\Pi : (q_t, K_t) \to \langle A_{i_1}, \ldots, A_{i_k} \rangle \times \text{Coord} \quad (5)$$

where Coord defines communication patterns, consistency verification, and conflict resolution.

3.3 Consistency Requirements

We enforce consistency at three levels:

Level 1 (Intra-Agent): Each agent's output satisfies its local constraints:

$$\forall A_i, \forall o \in \text{outputs}(A_i) : o \models C_i \quad (6)$$

Level 2 (Inter-Agent): Outputs from different agents must not contradict:

$$\forall A_i, A_j, \forall o_i \in \text{outputs}(A_i), o_j \in \text{outputs}(A_j) : o_i \wedge o_j \not\models \bot \quad (7)$$

Level 3 (Temporal): Outputs must remain consistent across dialogue turns:

$$\forall t_1 < t_2, o_{t_1} \wedge o_{t_2} \not\models \bot \quad (8)$$

4 Multi-Agent Architecture

Our architecture organizes 18+ specialized agents into a four-tier hierarchy designed to address the requirements from Section 3.

4.1 Tier 1: Perception and Observation

Perception agents operate asynchronously, continuously monitoring learner activity without blocking the request-response cycle:

Behavioral Analytics Agent Tracks engagement patterns including time-on-task, session frequency, topic revisitation, and learning velocity. Generates behavioral features used by downstream agents for adaptive scaffolding.

Error Pattern Miner Analyzes failed reasoning attempts to identify systematic misconceptions. Clusters similar error types (e.g., confusing necessary vs. sufficient conditions, quantifier scope errors) and builds taxonomies that inform intervention strategies.

Logical Consistency Monitor Validates Levels 2 and 3 consistency from Section 3. Flags contradictions between agent outputs and maintains temporal coherence across session history through automated logical inference.

These agents generate insights asynchronously, triggering proactive interventions without increasing response latency.

4.2 Tier 2: Domain-Specialized Agents

Domain agents provide deep expertise in specific technical areas through hierarchical sub-agent delegation:

Code Analysis Agent Performs symbolic reasoning over program semantics. Delegates to four sub-agents:

- **Syntax Validator:** Checks syntactic correctness via formal grammar rules
- **Type Checker:** Performs static type inference and constraint solving
- **Semantic Analyzer:** Evaluates program behavior through symbolic execution
- **Complexity Evaluator:** Analyzes algorithmic complexity via recurrence relations

System Design Agent Reasons about distributed systems through formal models.

Mathematical Reasoning Agent Performs symbolic manipulation and proof construction:

- **Algebraic Solver:** Symbolic equation solving and inequality reasoning
- **Proof Checker:** Validates mathematical proofs for logical soundness

The hierarchical structure enables parallel execution while maintaining coherent domain-level synthesis.

4.3 Tier 3: Coordination and Meta-Agents

Central Coordinator Orchestrates multi-agent reasoning through:

1. **Intent Classification:** Analyzes queries to determine required domains and reasoning types
2. **Context Propagation:** Maintains shared context including reasoning history, established facts, and accumulated constraints
3. **Response Aggregation:** Synthesizes outputs from multiple agents while maintaining consistency

Quality Assurance Meta-Agent Reviews all outputs before delivery.

4.4 Tier 4: Strategic and Long-Term Planning

Strategic agents operate on extended timescales, generating guidance that spans weeks to months:

Prediction Agent Forecasts learner trajectories including dropout risk (using engagement time-series analysis), time-to-mastery estimates, and struggle area identification before failures occur.

Intervention Recommender Generates proactive engagement strategies including personalized micro-lesson suggestions, difficulty adjustments to maintain flow state, and gamification triggers based on motivation profiles.

4.5 Agent Communication Protocol

Agents communicate through structured JSON payloads that enable explicit coordination:

```
{
  "query": <learner question>,
  "context": {
    "skill_level": <proficiency>,
    "history": <prior turns>,
    "constraints": <established facts>
  },
  "routing": {
    "primary_agents": [<domains>],
    "confidence": <score>
  }
}
```

The protocol supports parallel execution (independent agents process simultaneously), failure isolation (agent failures don't cascade), consistency verification (explicit checking at each coordination step), and complete explainability (full reasoning chains preserved in interaction traces).

5 Experimental Setup

5.1 Implementation

We implemented the architecture on a production learning platform with the following technical stack:

- **LLM Backend:** GPT-4 for primary agents and coordinator; GPT-3.5-turbo for subagents requiring lower latency
- **Vector Store:** Pinecone for semantic retrieval of relevant theorems, code patterns, and worked examples
- **Orchestration:** Custom Python framework managing agent lifecycle, communication, and consistency checking
- **Knowledge Base:** PostgreSQL storing skill graphs, engagement timeseries, error histories, and inference chains

5.2 Evaluation Methodology

We evaluate our system through two complementary approaches:

Benchmark Evaluation We assess logical reasoning capabilities on three established datasets:

- **ProofWriter** (27): Multi-step deductive reasoning from rule sets (metrics: proof correctness, step accuracy)
- **MATH** (13): High school mathematics requiring symbolic manipulation (metrics: answer accuracy, step correctness, hallucination rate)
- **LogiQA** (18): Logical reasoning with emphasis on consistency (metrics: overall accuracy, negation handling, contradiction detection)

Deployment Evaluation Metrics include session engagement, multi-turn dialogue depth, user-reported errors, and reasoning consistency.

5.3 Baselines

We compare against three baselines:

1. **GPT-4 Monolithic:** Standard GPT-4 with educational system prompts
2. **GPT-4 + Chain-of-Thought:** GPT-4 with explicit chain-of-thought prompting (31)
3. **GPT-4 + Program-Aided:** GPT-4 with tool integration for symbolic computation (12)

All baselines use identical underlying models; differences arise purely from architectural choices.

System	Proof Acc.	Step Acc.	Avg. Steps
ProofWriter Dataset			
GPT-4 Mono.	67.2%	71.4%	4.8
GPT-4 + CoT	74.5%	78.1%	5.2
Multi-Agent	**82.3%**	**86.7%**	**4.1**
MATH Dataset			
	Ans. Acc.	**Step Corr.**	**Halluc. Rate**
GPT-4 Mono.	52.4%	64.2%	18.3%
GPT-4 + Prog.	61.7%	72.5%	12.1%
Multi-Agent	**68.9%**	**79.8%**	**7.4%**
LogiQA Dataset			
	Overall Acc.	**Negation Acc.**	**Contradict. Detection**
GPT-4 Mono.	71.2%	58.3%	62.1%
GPT-4 + Self-C.	76.8%	64.7%	68.4%
Multi-Agent	**81.4%**	**73.2%**	**79.6%**

Table 1: Performance on logical reasoning benchmarks. Multi-Agent system shows consistent improvements across all metrics and datasets.

6 Results

6.1 Benchmark Performance

Table 1 presents results on logical reasoning benchmarks.

Our multi-agent system achieves substantial improvements across all benchmarks:

Deductive Reasoning (ProofWriter) 15.1% improvement in proof correctness over GPT-4 baseline, with higher intermediate step accuracy (86.7% vs. 71.4%) and more efficient proof construction (4.1 vs. 4.8 average steps).

Symbolic Reasoning (MATH) 7.2% improvement over program-aided baseline, with dramatic reduction in hallucination rate (7.4% vs. 18.3% for monolithic). The Mathematical Reasoning Agent's symbolic manipulation expertise provides clear benefits.

Logical Consistency (LogiQA) 4.6% overall improvement with particularly strong gains in negation handling (14.9% improvement) and contradiction detection (11.2% improvement), validating our consistency enforcement mechanisms.

6.2 Deployment Metrics

Table 2 presents results from production deployment serving thousands of learners over three months.

Metric	Baseline	Multi-Agent	Improv.
Avg. session (min)	28	43	+53.6%
Multi-turn exchanges	3.2	6.8	+112.5%
User-reported errors	14.7%	6.3%	-57.1%
Complex proofs done	41%	64%	+56.1%
Reasoning depth	3.1	5.7	+83.9%
Week-2 return rate	67%	81%	+20.9%

Table 2: Production deployment metrics comparing baseline (prior to multi-agent deployment) vs. multi-agent system.

Level	Mono.	Multi-Agent	Detect Rate	Resolve Rate
Intra-Agent	8.2%	2.1%	N/A	N/A
Inter-Agent	12.7%	4.3%	94.6%	87.3%
Temporal	18.4%	5.9%	89.2%	81.7%

Table 3: Consistency violation rates and resolution success. Multi-Agent system achieves 66.1% reduction in temporal inconsistencies with high detection and resolution rates.

The multi-agent architecture produces substantial engagement improvements: sessions increase from 28 to 43 minutes (+53.6%), multi-turn exchanges more than double (3.2 to 6.8), and user-reported logical errors decrease by 57.1%. The system enables 83.9% deeper reasoning chains (5.7 vs. 3.1 steps) while maintaining higher accuracy—suggesting that specialized agents successfully scaffold more sophisticated reasoning without overwhelming learners.

6.3 Consistency Maintenance

Table 3 analyzes consistency violations across 10,000 multi-turn reasoning sessions.

The multi-agent architecture achieves dramatic consistency improvements: 74.4% reduction in intra-agent inconsistencies (8.2% to 2.1%), 66.1% reduction in inter-agent violations, and 68.0% reduction in temporal inconsistencies. High detection rates (89-95%) and resolution rates (81-87%) validate our consistency monitoring and conflict resolution protocols.

7 Analysis and Emergent Behaviors

7.1 Why Multi-Agent Orchestration Works

We identify four mechanisms responsible for performance improvements:

Explicit Specialization Enables Expertise Monolithic models must simultaneously parse natural language, identify logical frameworks, perform symbolic manipulation, maintain inference chains, and verify consistency. Our Mathematical Reasoning Agent achieves 79.8% step correctness versus 64.2% for general-purpose GPT-4, demonstrating benefits of domain-focused optimization.

Coordination Protocols Enforce Consistency Monolithic models rely entirely on context window attention to prevent contradictions. Our explicit three-level consistency checking (Section 3) reduces temporal inconsistencies by 66.1%, with the Logical Consistency Monitor providing automated verification that supplements rather than replaces model capabilities.

Hierarchical Decomposition Reduces Cognitive Load Complex reasoning tasks overwhelm single models attempting to balance multiple objectives. Our hierarchical decomposition across four tiers enables 83.9% deeper reasoning (5.7 vs. 3.1 steps) while maintaining higher accuracy—suggesting that task decomposition prevents the cognitive overload observed in monolithic systems.

7.2 Observed Emergent Behaviors

We document three emergent coordination patterns that arose without explicit programming:

Cross-Domain Inference Chains When asked "Prove this sorting algorithm is correct and analyze its complexity": Mathematical Reasoning Agent constructs correctness proof via loop invariants, then Code Analysis Agent references the proof structure to guide complexity analysis. This cross-domain coordination emerges from agents accessing shared context.

Adaptive Proof Strategies The system automatically switches proof techniques based on problem structure: direct proof for straightforward implications (87% of cases), proof by contradiction when direct approaches stall (9%), and induction for recursive structures (4%). This adaptation emerges from agents observing each other's progress and adjusting strategies accordingly.

8 Discussion

8.1 Limitations

Our approach faces some of these limitations:

Coordination Overhead Multi-agent communication adds 1-2 seconds latency compared to monolithic baselines (average 3.2s vs. 1.4s). For real-time tutoring, this overhead may be prohibitive. Asynchronous execution and selective consistency checking partially mitigate this, but fundamental coordination costs remain.

Increased Engineering Complexity Deploying 18+ agents requires significantly more engineering than single-model systems. Prompt engineering at scale, debugging distributed reasoning chains, and managing agent versioning present operational challenges not present in monolithic architectures.

9 Ethical Considerations

Deploying AI agents in educational contexts raises important ethical concerns that require ongoing attention:

Algorithmic Bias Agent recommendations may perpetuate educational inequities if training data reflects historical biases. We implement continuous auditing for disparate impact across demographic groups, monitoring completion rates, engagement patterns, and recommendation quality across learner populations. Preliminary analysis shows no significant demographic differences, but sustained vigilance is required.

10 Conclusion

We presented a hierarchical multi-agent architecture for educational dialogue that addresses fundamental limitations in monolithic LLM-based tutoring systems. By decomposing complex educational interactions into specialized agent collaborations organized across four tiers—perception, domain expertise, coordination, and strategic planning—we demonstrate substantial improvements: 15.1% higher deductive reasoning accuracy, 66.1% reduction in temporal inconsistencies, and 83.9% deeper multi-turn reasoning chains in real-world deployment serving thousands of learners.

Our work demonstrates that the multi-agent paradigm offers a principled approach to educational dialogue systems, with implications extending to any complex NLP application requiring specialized expertise, consistency guarantees, and interpretable decision-making. The future of educational AI lies not in building larger monolithic models, but in orchestrating specialized agents that collectively achieve what no single model can—while providing transparency into how and why decisions are made.

Limitations

Latency Overhead: Multi-agent coordination adds 1-2 seconds compared to monolithic baselines, potentially prohibitive for real-time applications.

Engineering Complexity: Managing 18+ agents requires substantial engineering effort including prompt versioning, debugging distributed traces, and coordination protocol development.

Evaluation Gaps: Traditional benchmarks inadequately measure multi-agent coordination quality. New evaluation frameworks specifically targeting emergent behaviors, consistency maintenance, and interpretability are needed.

References

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160.

[2] John R. Anderson et al. 1995. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207.

[3] Alejandro Barredo Arrieta et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

[4] Rishi Bommasani et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[5] Tom Brown et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

[6] Jakub A. Chudziak and Arkadiusz Kostka. 2024. Leveraging large language models for personalized learning: An adaptive tutoring approach. *arXiv preprint arXiv:2407.12484*.

[7] Kevin Clark et al. 2019. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

[8] Karl Cobbe et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

[9] Albert T. Corbett and John R. Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278.

[10] Nouha Dziri et al. 2023. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*, volume 36.

[11] Jacques Ferber. 1999. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, Reading, MA.

[12] Luyu Gao et al. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 10764–10799. PMLR.

[13] Dan Hendrycks et al. 2021. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, volume 34, pages 13284–13297.

[14] Sirui Hong et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

[15] Enkelejda Kasneci et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

[16] Khan Academy. 2023. Khanmigo: Leveraging GPT-4 to supercharge learning. https://www.khanacademy.org/khan-labs. Accessed: 2024-12-15.

[17] Mark Liffiton et al. 2023. CodeHelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, pages 1–11, Koli, Finland. ACM.

[18] Jian Liu et al. 2020. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pages 3622–3628.

[19] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774.

[20] Jakub Macina et al. 2023. MathDial: A dialogue tutoring dataset with rich pedagogical properties grounded in math reasoning problems. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5878–5893, Singapore. Association for Computational Linguistics.

[21] Joon Sung Park et al. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, San Francisco, CA. ACM.

[22] Chris Piech et al. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, volume 28, pages 505–513.

[23] Chen Qian et al. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

[24] Marco Tulio Ribeiro et al. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, San Francisco, CA. ACM.

[25] Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

[26] Timo Schick et al. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551.

[27] Oyvind Tafjord et al. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

[28] Kurt VanLehn. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221.

[29] Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

[30] Yousef Wardat et al. 2023. ChatGPT: A revolutionary tool for teaching and learning mathematics. *Eurasia Journal of Mathematics, Science and Technology Education*, 19(7):em2286.

[31] Jason Wei et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

[32] Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, UK, second edition.

[33] Qingyun Wu et al. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

[34] Zheyuan Zhang et al. 2024. SimClass: Simulating classroom education with LLM-empowered agents. *arXiv preprint arXiv:2406.19226*.