International Journal of Advanced Multidisciplinary Application | IJAMA
Volume 2 Issue 1 Jan 2025
ISSN No: 3048-9350

A Comprehensive Survey on Agentic Artificial Intelligence Architectures: Reasoning, Planning and Real-World Applications

Dr. K. Sujatha
Independent Researcher

Abstract

Agentic Artificial Intelligence (AI) marks a fundamental paradigm shift: from passive language models to autonomous systems capable of multi-step reasoning, adaptive planning, dynamic tool use, and sustained goal-directed behaviour. This survey provides a structured review of the theoretical foundations, architectural patterns, reasoning strategies, and real-world deployments of agentic AI. We examine the core components of contemporary agentic frameworks — perception, reasoning engines, memory hierarchies, planning modules, and tool interfaces — drawing on recent literature spanning large language model (LLM)-based agents, multi-agent systems, embodied agents, and hybrid symbolic-neural approaches. Advanced reasoning paradigms including Chain-of-Thought (CoT), Tree-of-Thought (ToT), ReAct, Reflexion, and Monte Carlo Tree Search (MCTS)-augmented planning are analysed alongside the emerging challenges of alignment, safety, and scalability. Real-world applications across software engineering, scientific discovery, healthcare, robotics, enterprise automation, and education are surveyed, and critical open problems are identified. This survey is intended as a reference architecture for researchers and practitioners engaged in the design, evaluation, and deployment of agentic AI systems.

Keywords: *Agentic AI, large language models, autonomous agents, chain-of-thought, tree-of-thought, ReAct, multi-agent systems, planning, tool use.*

1. Introduction

The past several years have witnessed a remarkable transformation in artificial intelligence driven by large language models (LLMs) such as GPT-4, Claude, Gemini, and LLaMA. While early LLMs served as single-turn generation engines, subsequent research has shown that these models can act as the cognitive core of far more capable systems that autonomously plan multi-step sequences, use external tools, maintain persistent memory, and execute complex tasks across extended time horizons. These systems are collectively described as agentic AI, and their rapid maturation has opened domains previously inaccessible to machine intelligence.

Agentic AI differs from conventional machine learning in three critical respects. First, agentic systems operate in closed-loop environments, receiving feedback from actions and adjusting behaviour accordingly. Second, they are strongly goal-oriented: given a high-level objective, an agent must formulate sub-goals, allocate resources, select tools, and manage uncertainty across a temporal horizon that may span hundreds of individual steps. Third, agents must handle partial observability, gathering and synthesising heterogeneous information sources dynamically rather than reasoning from a fixed, complete context.

The intellectual lineage of agentic AI spans classical AI planning (STRIPS, PDDL), reinforcement learning, cognitive architectures (SOAR, ACT-R), autonomous robotics, and, most recently, the transformer-based LLM revolution. What distinguishes contemporary agentic AI from its predecessors is the integration of powerful pre-trained foundation models as general-purpose reasoning engines, enabling language understanding, common-sense inference, and cross-domain generalisation unattainable by earlier planning systems.

This survey is organised as follows. Section 2 describes canonical agentic AI architecture. Section 3 reviews principal reasoning and planning strategies. Section 4 examines memory and state management. Section 5 covers tool use and environment interaction. Section 6 surveys multi-agent coordination. Section 7 reviews real-world applications. Section 8 discusses open challenges. Section 9 concludes.

2. Architectural Foundations of Agentic AI Systems

2.1 Core Architectural Components

A canonical agentic AI system decomposes into three principal layers: a perception layer that processes inputs from the environment; a reasoning and planning layer that translates perceived state into intended action sequences; and an action execution layer that interfaces with external tools, APIs, databases, and physical actuators. As illustrated in Fig. 1, these layers interact through continuous feedback loops enabling adaptive replanning when execution outcomes deviate from expectations.

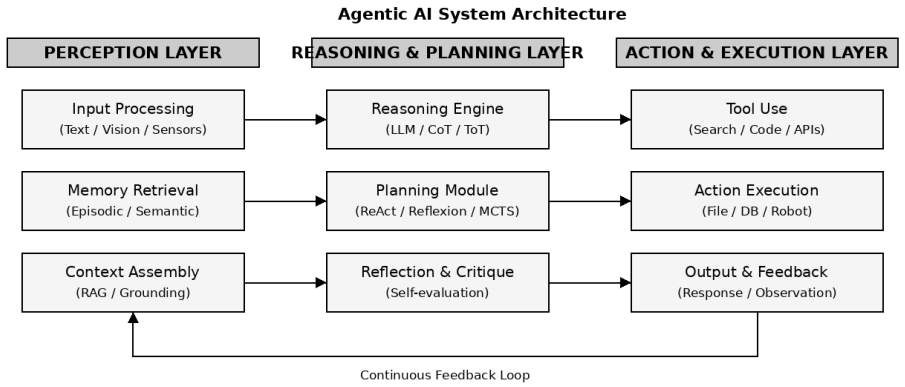

Fig. 1. Three-layer Agentic AI Architecture: Perception, Reasoning & Planning, and Action & Execution, connected by a continuous feedback loop.

The perception layer encompasses natural language (the primary LLM-agent modality), structured data, images, audio, and sensor streams in multimodal configurations. Modern agents increasingly leverage vision-language models such as GPT-4V, Gemini Pro Vision, and LLaVA to process visual information alongside text. Translating raw environmental signals into agent-legible representations — grounding — remains an active area of research, particularly in robotics where the gap between symbolic representation and physical reality is most pronounced.

The reasoning engine, typically instantiated as an LLM with system-prompt-defined role constraints, constitutes the central cognitive component. It receives observations, consults stored memories, evaluates available tools, and generates direct action outputs or intermediate reasoning traces. The quality of its inferences — decomposing ambiguous goals, detecting inconsistencies, recovering from planning failures — is the primary determinant of task success rates.

The action execution layer provides the interface between the agent's internal representations and the external world. For software-domain agents this means function-calling APIs through which the agent invokes web search, executes code, reads or writes files, and calls external services. For embodied agents the action space is defined by actuator capabilities and physical constraints. The design of this action space — its granularity, composability, and error semantics — profoundly influences agent robustness.

2.2 Memory Architecture

Memory in agentic AI encompasses four functionally distinct types. Working memory, analogous to the LLM's context window, holds the current observation, task state, and recent action history. Episodic memory stores records of past agent episodes and supports retrieval-augmented generation (RAG). Semantic memory encodes factual world knowledge, increasingly augmented by domain-specific knowledge bases. Procedural memory captures learned action policies and reusable action sequences, enabling agents to develop and recall generalised skills.

Memory retrieval — determining which past experiences are most relevant to the current context — is typically implemented via dense vector search over embedding representations using systems such as Weaviate, Pinecone, or ChromaDB. Memory consolidation — deciding which experiences to retain, compress, or discard — is critical for long-horizon operation, where unbounded context accumulation leads to performance degradation and increased computational cost.

3. Reasoning Strategies in Agentic AI

3.1 Chain-of-Thought and Variants

Chain-of-Thought (CoT) prompting [1] demonstrated that instructing LLMs to generate intermediate reasoning steps before producing a final answer substantially improves multi-step reasoning performance. In the agentic context, CoT

serves as the agent's internal monologue — a sequence of steps translating high-level goals into concrete action specifications. Standard CoT operates sequentially and is vulnerable to early errors that propagate undetected through the chain.

Tree-of-Thought (ToT) [2] addresses this by framing reasoning as search over a tree of possible reasoning continuations. At each step, the model generates multiple candidate next-thought expansions, evaluates their promise, and selects the most promising branch. ToT yields particularly strong results on planning-intensive tasks — multi-step arithmetic, creative writing with hard constraints, and logical puzzle solving — at the cost of significantly higher inference expenditure.

3.2 ReAct, Reflexion, and MCTS Planning

ReAct [3] represents a landmark contribution by proposing explicit interleaving of reasoning traces and action outputs. Agents generate alternating thought–action–observation triplets: a natural language reasoning trace, a tool call, and incorporation of the resulting observation into the next reasoning step. This interleaving enables dynamic plan adjustment in response to tool outputs, substantially improving performance on interactive environments including web navigation and database querying.

Reflexion [4] extends this paradigm by introducing explicit self-reflection: after completing a task episode, the agent generates a natural language critique identifying specific errors and formulating improved strategies for future attempts. These verbal reflections are stored in episodic memory and retrieved in subsequent attempts, enabling gradient-free learning from experience. Reflexion achieves state-of-the-art results on AlfWorld and HotpotQA without any weight updates to the underlying LLM.

Monte Carlo Tree Search (MCTS)-augmented agents treat planning as search over action sequences, using the LLM as both a policy network and a value function. This integration allows deeper lookahead than greedy ReAct-style approaches while remaining grounded in natural language. Fig. 2 and Table 1 compare the empirical accuracy of these principal reasoning strategies on complex multi-step benchmarks.

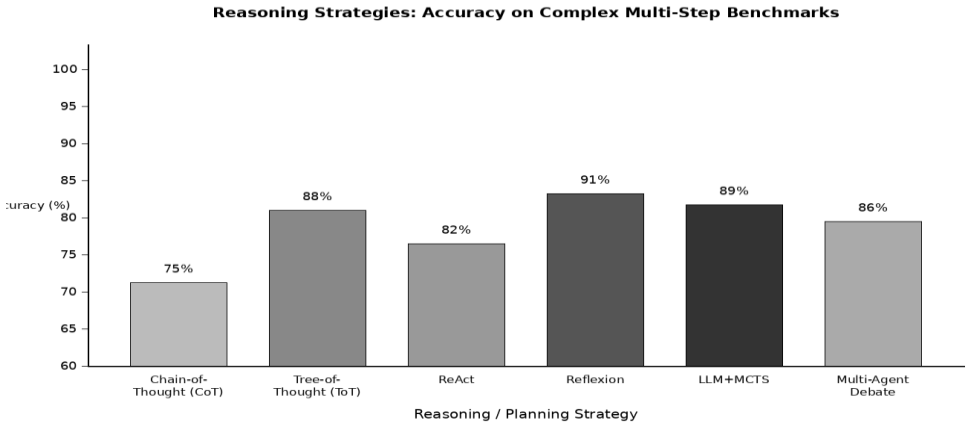

Fig. 2. Benchmark accuracy (%) of key agentic AI reasoning strategies on complex multi-step tasks. Reflexion achieves the highest accuracy (91%) through iterative self-correction.

Table 1. Comparative Summary of Agentic AI Reasoning Strategies

Method	Mechanism	Best Use Case	Limitation
Chain-of-Thought (CoT)	Step-by-step reasoning trace	Math, QA, logic	Error propagation in long chains
Tree-of-Thought (ToT)	Branch-and-prune reasoning	Complex planning	High inference cost
ReAct	Reasoning + action interleaved	Tool-use tasks	Dependent on tool reliability
Reflexion	Verbal self-reflection loop	Iterative optimization	Convergence not guaranteed

Method	Mechanism	Best Use Case	Limitation
MCTS-augmented Planning	Monte Carlo tree search	Strategic & game tasks	Scalability challenges
LLM+P (PDDL planner)	LLM generates PDDL; planner solves	Structured domains	Formal domain modelling burden

3.3 Classical Planning Integration

The LLM+P framework [7] combines LLMs with classical AI planners operating over Planning Domain Definition Language (PDDL). An LLM translates natural language task specifications into formal PDDL problem files, which are solved by a classical planner such as Fast Downward or FF. The resulting plan is translated back into natural language for execution. This approach leverages the complementary strengths of LLMs (language understanding, common-sense knowledge) and classical planners (soundness guarantees, optimal solution generation), while the principal limitation is the difficulty of constructing accurate formal domain models for open-world tasks.

4. Tool Use and Environment Interaction

Tool use is arguably the capability that most distinguishes agentic AI from conventional LLMs. It allows agents to overcome static knowledge limitations by dynamically accessing current information via web search, executing precise computations through code interpreters, manipulating persistent data in file systems and databases, and interacting with external services through APIs. The design of the tool interface — which tools are available, how they are described, how errors are communicated, and how outputs are incorporated — profoundly affects performance.

Function calling, as implemented in OpenAI's GPT-4 API, Anthropic's tool-use interface, and Google's Gemini API, provides a standardised mechanism for structured tool invocations. The model generates a typed tool call specification executed by a runtime layer, with results returned as subsequent context entries. This cleanly separates reasoning from execution semantics, enabling modular capability extension without retraining. The principal challenge is tool selection and argument grounding: correctly identifying the appropriate tool and constructing valid arguments, both of which degrade as the number of available tools increases.

Toolformer [15] demonstrated that LLMs can learn to use tools (web search, calculators, calendar APIs, translation) through self-supervised training, using a small number of API call demonstrations to generate training data at scale. This self-supervised paradigm contrasts with manual tool-use instruction and suggests a scalable pathway to expanding agent tool repertoires. Code execution represents a particularly powerful tool primitive: a Turing-complete code interpreter subsumes many other tool capabilities, and Code as Policies [21] demonstrated that code-based action representations generalise well to robotics manipulation planning.

5. Multi-Agent Systems and Coordination

5.1 Multi-Agent Architectures

Many real-world tasks benefit from multi-agent architectures in which multiple AI agents with specialised roles collaborate toward a shared objective. Multi-agent systems can distribute cognitive labour across agents (parallelising independent subtasks), enable adversarial self-improvement through debate and critique, and provide redundancy through cross-agent verification. AutoGen [5] provides an abstraction for defining agent roles, communication protocols, shared memory resources, and task orchestration logic. In a typical configuration, an orchestrator receives the high-level task, decomposes it into subtasks, delegates to specialised worker agents (researcher, coder, critic), and synthesises outputs into a coherent final result.

5.2 Communication and Consensus

Inter-agent communication in LLM-based multi-agent systems is primarily conducted in natural language, which offers interpretability but introduces ambiguity and potential for miscoordination. Multi-agent debate frameworks [16] address consensus through iterative rounds of argument and counterargument, converging to a consensus position empirically more accurate than individual agent outputs on factual reasoning tasks. Generative Agents [17] demonstrated that populations of LLM-based agents in simulated social environments exhibit emergent behaviours — spreading information, forming opinions, coordinating activities — analogous to human social dynamics, providing a compelling proof-of-concept for LLM-based multi-agent social simulation.

6. Planning in Long-Horizon Tasks

Long-horizon planning — pursuing goals requiring hundreds or thousands of sequential decisions — remains one of the most challenging problems in agentic AI. Current LLM-based agents exhibit several documented failure modes: context window saturation, compounding error propagation, goal drift over extended episodes, and reward hacking by discovering unintended shortcuts. Hierarchical planning addresses context saturation by decomposing tasks into multiple levels of abstraction, maintaining high-level plans in compressed symbolic representations while expanding only the current sub-goal into detailed action sequences.

VOYAGER [6] demonstrated lifelong learning in the Minecraft environment through a curriculum module proposing progressively challenging sub-goals, a code generation module producing skill implementations, and a skill library accumulating reusable programmatic primitives that extend the effective planning horizon over time without growing the active context window. The framework achieved continual acquisition of new skills — from basic tool crafting to complex automated farm construction — without any fine-tuning of the underlying LLM, illustrating the power of hierarchical agentic architecture for open-ended task domains.

Sub-goal generation and progress estimation — the capacity to assess how much of the overall task has been completed and what remains — are critical supporting capabilities for long-horizon planning that receive comparatively little attention in the literature. As agentic AI systems are deployed on increasingly extended real-world workflows, robust progress estimation and dynamic replanning will become essential engineering requirements alongside the foundational reasoning capabilities reviewed above.

7. Real-World Applications

7.1 Software Engineering

Software engineering is among the most extensively studied application domains. The SWE-bench benchmark [9] evaluates agents on real GitHub issue resolution tasks; early baselines achieved resolution rates below 5%, while recent systems incorporating agentic scaffolding, code execution environments, and multi-file context management have surpassed 50%. The Devin system [25] demonstrated end-to-end autonomous software engineering including requirements analysis, implementation, test writing, debugging, and deployment in controlled evaluations. The architectural pattern common to high-performing systems involves a sandboxed code execution environment, file system tools for reading and modifying source files, test execution tools for validating changes, and a version control interface for managing code state.

7.2 Scientific Discovery

The AI Scientist framework [11] demonstrated an end-to-end autonomous research pipeline generating research ideas from a seed topic, reviewing related literature, writing experimental code, executing experiments, analysing results, and drafting manuscript submissions — producing outputs human evaluators rated comparable to workshop paper quality. ChemCrow [10] integrated an LLM with 18 chemistry tools including reaction planners, molecular property predictors, and retrosynthetic analysis engines, enabling autonomous synthesis route planning for target molecules.

7.3 Healthcare and Clinical Decision Support

Med-PaLM 2 [14] achieved expert-level performance on medical licensing examinations, but clinical deployment requires agentic capabilities — retrieving patient history from electronic health records, ordering additional investigations, synthesising evidence from multiple sources, and generating differential diagnoses with supporting reasoning traces. Agentic frameworks applying multi-step reasoning to clinical case presentations have demonstrated diagnostic accuracy competitive with resident physicians on structured evaluation benchmarks, while challenges of regulatory compliance, liability attribution, and distribution shift under real-world clinical conditions remain largely unresolved.

7.4 Robotics and Embodied Agents

RT-2 [12] demonstrated that vision-language-action models trained on large-scale internet data can generalise to novel manipulation tasks not present in the training distribution, exhibiting emergent physical reasoning absent from models trained on robot-specific data alone. PaLM-E [13] further showed that embodied planning performance scales with model size analogously to language task performance, suggesting that the same scaling laws governing LLM capability generalise to the embodied domain. These results imply that continued scaling of foundation models will yield continued improvement in robotic agent capabilities, contingent on sufficient embodied experience data.

7.5 Enterprise Automation and Education

Enterprise agentic AI applications encompass document processing, workflow automation, customer service, financial analysis, and supply chain management. Microsoft Copilot Studio, Salesforce Agentforce, and Google's Vertex AI Agent

Builder provide production-grade agentic infrastructure with guardrails including access control, audit logging, and human-in-the-loop approval workflows for high-consequence actions. In education, agentic tutoring systems such as Khanmigo apply multi-step Socratic dialogue and adaptive feedback to personalised learning, with early evidence of improved student engagement and learning outcomes compared to static content delivery.

8. Safety, Alignment, and Open Challenges

8.1 Alignment and Value Specification

As agentic AI systems take increasingly consequential actions, alignment — ensuring agent behaviour reflects the intentions and values of the humans directing it — becomes correspondingly critical. Natural language system prompts are inherently ambiguous: instructions rarely cover all contingencies arising during extended task execution, and agents must fill gaps through inference that may not match human intent. Prompt injection attacks — adversarial inputs embedded in the environment (e.g., in web pages or documents retrieved during task execution) that hijack the agent's action policy — represent a particularly concerning attack surface unique to agentic deployments with no robust mitigation yet established.

Constitutional AI [8] and related reinforcement learning from human feedback (RLHF) approaches instil value-aligned behavioural dispositions in the foundation model, reducing the likelihood that an agent pursues harmful sub-goals even under ambiguous task specifications. However, these approaches operate at the level of individual model outputs and do not directly address multi-step planning failures that characterise long-horizon agentic misbehaviour. Formal specification of agent objectives in verifiable representations, and runtime monitoring of plan execution against safety constraints, represent important research directions.

8.2 Evaluation Methodology

Evaluating agentic AI presents methodological challenges not encountered in standard NLP benchmarks. Task success is often binary, masking reasoning quality and providing limited diagnostic signal. Intermediate-step evaluation — assessing individual reasoning steps rather than only final outcomes — requires annotated execution traces expensive to produce at scale. Reproducibility is compromised by stochastic LLM inference and dynamic execution environments. Benchmarks including GAIA, AgentBench [18], WebArena [19], and OSWorld have advanced the field significantly, but coverage of long-horizon, multi-agent, and safety-critical scenarios remains limited, and there is no consensus on evaluation protocols for assessing agent alignment properties.

8.3 Computational Efficiency

Agentic systems employing ToT, MCTS, or multi-agent configurations incur computational costs that scale unfavourably with task complexity. A single complex task may require dozens of LLM inference calls, consuming significant compute and incurring latency incompatible with real-time operation. Inference efficiency techniques including speculative decoding, KV-cache sharing across agent turns, capability distillation into smaller models, and mixture-of-experts architectures provide partial mitigations without eliminating the fundamental cost-performance trade-off. Hardware acceleration through custom silicon and optimised serving frameworks (vLLM, TensorRT-LLM) are increasingly important for production agentic deployments at scale.

9. Future Research Directions

Several high-priority research directions emerge from this survey. Robust long-horizon planning — maintaining goal coherence and recovering from compounding errors across task horizons of thousands of steps — remains an unsolved fundamental problem requiring advances in hierarchical planning, memory consolidation, and progress estimation alongside improved foundation model capabilities. Principled multi-agent coordination — moving beyond ad hoc natural language communication to structured protocols with formal correctness guarantees — is critical for deploying multi-agent systems in safety-critical domains.

Agentic AI safety — developing comprehensive threat models, evaluation frameworks, and mitigation strategies for the novel failure modes introduced by autonomous multi-step AI action — requires urgent research investment as agentic systems reach consequential real-world deployments. The adversarial robustness of agentic systems to prompt injection, goal hijacking, and indirect instruction attacks is particularly under-studied relative to the pace of deployment. Efficient agentic inference — reducing computational cost without proportionate performance degradation — will be essential for broad access across resource-constrained settings, with test-time compute scaling, adaptive reasoning depth, and learned planning heuristics offering promising directions.

10. Conclusion

This survey has presented a comprehensive review of agentic artificial intelligence — systems that move beyond passive response generation to autonomous, goal-directed reasoning and action across complex, extended tasks. We have examined the architectural foundations of agentic systems, surveyed principal reasoning and planning strategies, reviewed memory and tool-use mechanisms, characterised multi-agent coordination frameworks, and mapped real-world applications from software engineering and scientific discovery to healthcare, robotics, and enterprise automation.

The field is advancing at extraordinary pace: capabilities that required specialised research prototypes recently are now available through production APIs, and agentic AI systems are beginning to take consequential actions across an expanding range of domains. This trajectory underscores the urgency of the open challenges identified here — particularly alignment, safety, and long-horizon reliability — and the importance of rigorous evaluation frameworks that can keep pace with capability advances. Agentic AI is transitioning from research novelty to foundational infrastructure; developing principled, safe, and reliable agentic systems is therefore one of the most important challenges in contemporary computer science.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022, pp. 24824–24837.

[2] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[3] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proceedings of the International Conference on Learning Representations (ICLR), 2023.

[4] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[5] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv preprint arXiv:2308.08155, 2023.

[6] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "VOYAGER: An open-ended embodied agent with large language models," arXiv preprint arXiv:2305.16291, 2023.

[7] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: Empowering large language models with optimal planning proficiency," arXiv preprint arXiv:2304.11477, 2023.

[8] Y. Bai, S. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan et al., "Constitutional AI: Harmlessness from AI feedback," arXiv preprint arXiv:2212.08073, 2022.

[9] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "SWE-bench: Can language models resolve real-world GitHub issues?" arXiv preprint arXiv:2310.06770, 2023.

[10] A. M. Bran, S. Cox, A. D. White, and P. Schwaller, "ChemCrow: Augmenting large-language models with chemistry tools," arXiv preprint arXiv:2304.05376, 2023.

[11] C. Lu, C. Lu, R. T. Q. Chen, J. Hernandez-Garcia, M. Watter, and Y. Bengio, "The AI Scientist: Towards fully automated open-ended scientific discovery," arXiv preprint arXiv:2408.06292, 2024.

[12] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu et al., "RT-2: Vision-language-action models transfer web knowledge to robotic control," arXiv preprint arXiv:2307.15818, 2023.

[13] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu et al., "PaLM-E: An embodied multimodal language model," in Proceedings of the International Conference on Machine Learning (ICML), 2023.

[14] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl et al., "Large language models encode clinical knowledge," Nature, vol. 620, no. 7972, pp. 172–180, 2023.

[15] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[16] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," in Proceedings of the International Conference on Machine Learning (ICML), 2024.

[17] J. S. Park, J. C. O`Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in Proceedings of the ACM Symposium on User Interface Software and Technology (UIST), 2023.

[18] S. Zhang, J. Chen, Z. Shen, X. Chen, X. J. Zhu, and J. Zheng, "AgentBench: Evaluating LLMs as agents," arXiv preprint arXiv:2308.03688, 2023.

[19] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, Y. Bisk, D. Fried, U. Alon, and G. Neubig, "WebArena: A realistic web environment for building autonomous agents," arXiv preprint arXiv:2307.13854, 2023.

[20] R. Nakano, J. Hilton, S. Balwit, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Gray, A. Radford, and J. Schulman, "WebGPT: Browser-assisted question-answering with human feedback," arXiv preprint arXiv:2112.09332, 2021.

[21] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 9493–9500.

[22] M. Minsky, The Society of Mind. New York, NY: Simon & Schuster, 1986.

[23] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace," in Advances in Neural Information Processing Systems (NeurIPS), vol. 36, 2023.

[24] H. Chase, "LangChain: Building applications with LLMs through composability," GitHub repository, 2022. [Online]. Available: https://github.com/langchain-ai/langchain

[25] Cognition AI, "Introducing Devin: The first AI software engineer," Technical Report, Cognition AI, Mar. 2024.