EDITOR: Bo An, boan@ntu.edu.sg

COLUMN: AI'S 10 POSITION PAPERS

From Symbols to Synapses: The Reemergence of Agency in the Large Language Model Era

Bo An, *Nanyang Technological University, 6397987, Singapore*

Building intelligent agents capable for autonomously perceiving, reasoning, and acting to achieve goals has been a central pursuit of artificial intelligence (AI) since its inception. For decades, the notion of agency was dominated by symbolic architectures that represent information as formal knowledge and use deliberative reasoning to derive rational actions, offering reliability but at the cost of brittleness and limited generalizability. The recent advancements in large language models (LLMs) and their integration into tool-using, environment-interacting "agentic" systems have reignited interest in AI agents. However, while LLM-based agents provide the flexibility that symbolic systems lacked, they introduce new challenges in reliability and control. We posit that the future of AI agents lies not in indefinitely scaling the model size, but in synthesizing the methods and theories developed by the autonomous agents and multiagent systems community with modern neural architectures to create neuro-symbolic agents capable of trustworthy autonomy.

The concept of "intelligent agent," i.e., an autonomous entity that perceives inputs from environments, reasons over trajectories, and takes actions to achieve its goals, has been a central pursuit of artificial intelligence (AI) since its inception in the mid-20th century,[38] and later has been intensively studied by the autonomous agents and multiagent systems (AAMAS) community. In the early decades, the notion of agency was dominated by symbolic architectures, where information is represented as formal knowledge (e.g., first-order logic,[9] production rules,[27] and early formalisms, such as STRIPS[10]) and agents leverage deliberative reasoning to derive rational actions. Subsequent work introduced more expressive and solver-friendly formalisms, such as constraint-based encodings,[11,12] answer-set programs,[2] and planning languages like PDDL.[14] Such techniques and formalisms led to the first wave of progress in agency, including early general problem-solving architectures, such as the General Problem Solver (GPS),[26] and later expert systems like MYCIN[4] and the famous Deep Blue[16] that defeated a reigning world chess champion in 1997.

Unfortunately, manually encoding the infinite variability of real-world scenarios into formal rules before performing deliberation proved to be prohibitively difficult. As a result, the late 1990s saw a shift away from symbolic agency toward statistical learning and reinforcement learning (RL),[47] where the focus moved from deliberatively reasoning about rational plans and actions to finding the policies that maximize reward functions. TD-Gammon[49] is a notable example of RL agents, which leverages temporal-difference learning to reach human master level in backgammon. Beyond board games, RL was also successfully deployed in a

Digital Object Identifier 10.1109/MIS.2026.3668397
Date of current version 12 April 2026.

variety of sequential decision-making problems, such as elevator group control,[7] autonomous helicopter flight,[28] and RoboCup soccer.[46]

However, classical RL largely relies on tabular representations or relatively simple function approximators, which limits its ability to cope with raw, high-dimensional sensory inputs. The rise of deep learning in the 2010s catalyzed a new wave of deep RL (DRL) agents, in which deep neural networks are used as powerful function approximators for value functions and policies. Landmark work, such as the Deep Q-Network that achieved human-level performance on Atari 2600 games directly from pixels,[24] and AlphaGo that mastered the game of Go with deep policy/value networks,[44,45] demonstrated that end-to-end training from perception to action could scale to complex domains previously out of reach. Subsequent advances, including deep actor-critic methods[18,41,42] and multiagent deep RL frameworks,[19,37] further broadened the applicability of RL to continuous control, cooperative, and competitive multiagent settings.

Despite their impressive empirical successes, DRL agents are trained from scratch for specific tasks, which is not only time-consuming but also severely limits the capability of generalizing to novel or unknown tasks. The recent emergence of large language models (LLMs)[3,54] has opened a new paradigm for agency. Trained on web-scale corpora with autoregressive objectives, LLMs acquire broad world knowledge and emergent capabilities in reasoning, question answering, and code generation as model size scales. By integrating with scaffolds like tool use, memory, and environment interaction, LLM-based AI agents demonstrate stellar performance on various challenging tasks, such as software engineering,[35,55] computer control,[15,48] and AI for science[20,23,29] without costly task-specific retraining. An illustration of the evolution of agency is displayed in Figure 1.

LLM-BASED AGENTS

LLM-based agents[21,34] use LLMs as the core of a closed perception–reasoning–action loop. At each step, the agent receives a (typically natural language) description of the current state and goal, together with auxiliary context retrieved from memory or tools. The LLM then deliberates and selects an action, such as calling an external tool or executing code, after which the environment returns observations that are appended to the context for the next decision. In this view, the LLM functions as a planner or policy over textual states, with agency expressed in natural language rather than through explicit symbolic algorithms or purely learned control policies. Specifically, a typical LLM-based agent consists of:

- a planner LLM that serves as the reasoning and decision-making module
- an interface that mediates the interactions with external environments [e.g., application programming interfaces (APIs), code interpreters, computers, or tools]
- a memory that maintains both short-term working context for the current task and longer-term information such as user specifications, past experiences, or acquired skills.

Planner and Control Loop

Planner is the central control module in the perception–reasoning–action loop of LLM-based agents. The LLM is treated as a goal-conditioned policy over textual descriptions of state, deciding at each step whether to decompose the task, call a tool, request more information, or terminate.[21,34] It interprets the current state, the goal, and retrieved context, and produces the next action or intermediate subgoal. This mechanism, however, is far from robust. Planner outputs are stochastic and uncalibrated, long reasoning chains accumulate errors and hallucinations, and plans may violate safety or user constraints[34]; at the same time, limited context windows, underspecified goals, and ad hoc prompt engineering make behavior brittle and difficult to analyze. To mitigate these issues, current systems encode planning primarily in prompts (e.g., chain-of-thought,[52] ReAct[57]) and wrap the LLM in an explicit loop that alternates "think–act–observe" steps, while more structured variants perform search over LLM-generated thoughts,[56] use self-consistency or self-critique for verification,[43,51] and apply reinforcement learning to bias the planner toward safer and more goal-aligned trajectories.[31]

Interface With External Environments

The interface with external environments connects semantic outputs of LLMs to rigid external systems, such as APIs, code interpreters, or simulators, thereby turning LLM-generated actions into concrete calls and feeding the observations obtained from environments back to the textual context.[34,40] Given a planner's output, the interface selects the appropriate tool, validates arguments, executes the call, and formats the result into a concise description suitable for the next reasoning step, while also exposing environment

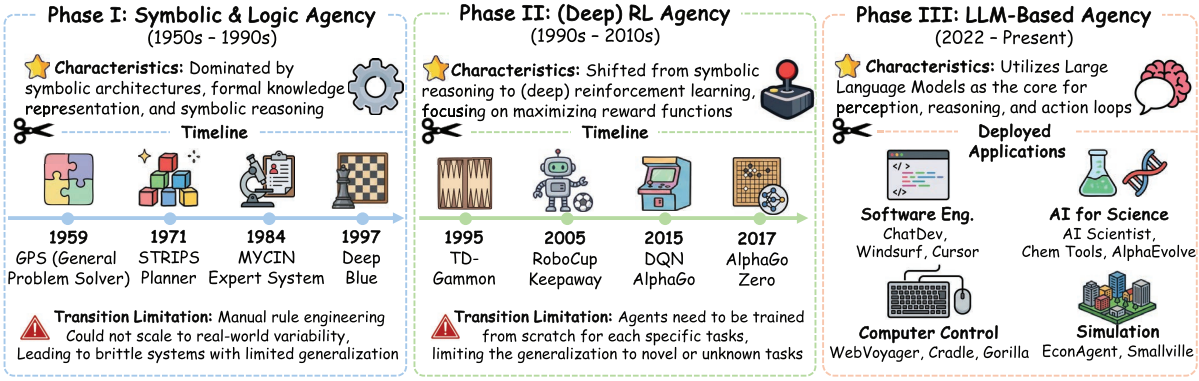

FIGURE 1. The evolution of agency from symbolic to modern LLM-centric architectures over the past decades.

state (files, databases, web, devices) as snippets that can be appended to the dialogue history. Designing this layer is also challenging: One must handle robust parsing and tool selection under free-form language, defend against malformed or unsafe actions (e.g., arbitrary code execution), and cope with latency, failures, or nondeterminism in external systems.[40] As the number and diversity of tools grow, choosing the right one and summarizing noisy outputs becomes increasingly difficult. In response, most LLM-based agents adopt function-calling-style prompting and tool schemas that specify names, arguments, and documentation, enabling the LLM to generate valid calls.[40] Frameworks, such as LangChain, encapsulate execution and error handling behind adapters, specialized models for API calling, or code execution are fine-tuned on synthetic tool-use corpora,[33] and many agentic architectures further constrain the planner through ReAct-like formats where the model interleaves natural-language "thoughts" with discrete tool invocations,[57] making the interface an integral part of the control loop.

Memory and Experience

Memory provides persistence beyond the planner's short-term context, storing user preferences, world knowledge, and the agent's own past trajectories, thus turning the stateless LLM into the agent that can accumulate experience, personalize behavior, and handle extended tasks that exceed a single context window.[58] Memory modules record episodes (dialogue turns, actions, outcomes) as well as more abstract summaries or skills, and expose them through retrieval at decision time, supporting both episodic recall and semantic recall.[36,58] The main design questions are what to store, at what granularity, and how to retrieve selectively. Simple accumulation leads to unbounded logs and expensive prompts, whereas aggressive pruning or summarization risks losing important details.[58] In addition, retrieval over large memories requires efficient indexing and relevance scoring, and the stored information can become stale or misaligned with evolving user preferences. Consequently, most agentic frameworks adopt retrieval-augmented memory, where interactions are embedded into a vector space and indexed in an external database, with top-k relevant items injected into the prompt on demand.[36] Systems then differ in how they segment and score memories (raw turns versus hierarchical summaries, recency, and importance weighting) and in how they combine these with short-term "working memory" inside the context window,[58] while some work goes further and uses fine-tuning or lightweight parameter editing to encode stable knowledge directly into the model, trading flexible retrieval for longer-term parametric adaptation.

DEPLOYMENT OF LLM-BASED AGENTS

As the capabilities of LLMs continue to improve, LLM-based agents are emerging as a key paradigm for deploying general-purpose models to concrete tasks. On the one hand, they integrate perception, reasoning, and action, updating internal state, and making decisions in pursuit of long-term objectives in complex environments. On the other hand, they are embedded in real-world environments, operating over extended periods in production code repositories, scientific experimental platforms, and office desktops. Compared with traditional AI systems that primarily optimize for specific tasks, LLM-based agents directly reflect how contemporary AI is actually used in practice with generalizability and adaptability. This section surveys

several representative applications of LLM-based agents to sketch the main contours and emerging trends of agentic AI in the real-world deployment.

Software Engineering

LLM-based agents in software engineering are evolving from function-level code generation into autonomous developers that can process long-horizon, repository-level tasks. "Agentic IDEs," such as Cursor, maintain built-in agents that can traverse a repository, execute shell commands, and coordinate multifile refactoring from within the editor interface.[8] Windsurf places particular emphasis on long-horizon "flows": The agent maintains state across many compilation and testing cycles, adjusting its plan as new errors and signals arise.[5] Devin, described as an "AI software engineer," runs in a sandbox that includes its own browser, terminal, and editor, and has been demonstrated on end-to-end tasks that require understanding tickets, modifying code bases, and submitting patches back into existing workflows.[6] In parallel, research platforms, such as SWE-agent, expose standardized interfaces to real GitHub repositories.[55] They allow LLM-based agents to browse, edit, and test code in open source projects and serve as benchmarks and reference implementations for this class of systems.

AI for Science

In scientific domains, the agentic use of LLMs is most apparent where systems are asked not only to make predictions, but also to help carry out the scientific research processes. One prominent line of work is the "AI scientist" paradigm, exemplified by Sakana AI's system that automates substantial portions of the research loop: Given a loosely specified topic, it can search and summarize literature, propose hypotheses, generate and modify code, run experiments, analyze results, and draft structured research reports.[20] Related efforts argue more broadly for "agentic" scientific workflows where AI systems take actions, such as selecting which experiments to run next, deciding which baselines to compare against, or organizing the structure of a study.[53] A complementary development is the rise of autonomous or "self-driving" laboratories in chemistry and materials science, where an AI agent chooses experimental conditions, controls robotic hardware, and updates internal models based on measured outcomes.[50] In many such systems, the LLM plays the role of planner and coordinator, translating high-level goals into experimental processes and orchestrating domain-specific solvers and optimizers. In more specialized settings, materials-design agents combine LLM planners with physics-informed simulation tools and curated databases to propose and evaluate candidate materials with target properties, iterating automatically over large-scale design spaces that would be infeasible to explore manually.[22]

Computer Control and Tool Use

A primary environment for LLM-based agents is the computer itself. Instead of using structured APIs, recent LLM-based agents perceive screens and windows and act via keyboard and mouse events. For example, OpenAI's computer use agents receive visual observations of an application window or browser, issue low-level interaction commands, and are trained to complete multistep tasks, such as filling out forms, booking travel, and aggregating information across multiple sites.[30] Anthropic's Claude "Computer Use" mode similarly runs as a remote worker on a user's desktop, combining GUI operations and command-line tools to execute long-running workflows that would previously have required human assistants.[1] Underpinning these systems is a rapidly developing infrastructure for structured tool use, including function-calling interfaces and the Model Context Protocol, which standardizes how external tools and data sources are exposed to LLM-based agents and discovered at run time.[25] Computer use agents and tool-use protocols are thus mutually synergizing: Protocols provide safe, auditable channels for tool use, while agents supply the decision to determine which tools to invoke.

Agent-Based Modeling

Early work on "generative agents" demonstrated that LLM-powered characters in a simple sandbox environment could maintain memories, form plans, and exhibit plausible emergent social behaviors over extended time horizons.[32] Building on this idea, several frameworks represent households, firms, or other actors with LLM-based agents whose micro-level decisions about work, consumption, saving, or communication collectively generate macro-level dynamics. For example, EconAgent and related systems use LLM-based populations to study macroeconomic phenomena and policy interventions, suggesting that LLM-based decision rules can reproduce patterns that are difficult to capture with simple hand-crafted behaviors.[17] In epidemiology and public health, multiagent frameworks that integrate LLM-based individuals with traditional compartmental or network models are being explored as "virtual policy sandboxes" in which alternative communication or intervention

strategies can be tested before real-world deployment.[39] Recent surveys refer to this broad line of work as "LLM-empowered agent-based modeling," and emphasize both its potential for computational social science and the challenges of calibration, validation, and ethical use that arise when simulated agents are powered by large pretrained models rather than explicitly designed rules.[13]

CONCLUSION AND OUTLOOK

The notion of agency has undergone a profound transformation over the past decades, evolving from symbolic architectures to deep neural models, and arriving at a modern renaissance where LLMs serve as core engines for perception, reasoning, and action. While this reemergence brings unprecedented generality and adaptability, LLM-based agents also introduce significant challenges regarding hallucinations, alignment, and long-horizon planning. Below we outline the critical open challenges that must be addressed to advance the field of LLM-based agents, and highlight the key opportunities where the AAMAS community is uniquely positioned to contribute in the era of LLMs.

Open Challenges in LLM-Based Agents

Long-Horizon Planning and Grounding

LLM-based agents demonstrate critical limitations when applied to long-horizon tasks involving extensively many steps (e.g., >1000 steps) or extended temporal durations (hours to days). This primarily stems from the fact that LLMs are trained to predict the next token based on statistical patterns in large text corpora, not to explicitly model the underlying dynamics of environments or causal logics. Besides, the internal world knowledge of LLMs may be incomplete, out-of-date, or misaligned with real environments, leading to compounding errors when decisions must be chained over long horizons. As a result, agents may produce plans that are semantically plausible but functionally invalid, due to violating implicit constraints, hallucinating unavailable capabilities, or overlooking environmental preconditions.

Memory

Effective agency requires a coherent memory of the learned skills, experiences, and working context for the problem currently being solved. Current memory architectures for LLM-based agents still remain primitive along three dimensions. First, pattern-matching approaches (cosine similarity, BM25, or LLM-based selection) decouple retrieval from decision making, retrieving semantically similar content while missing causal dependencies crucial for reasoning in decision-making tasks. Second, heuristic, rule-based mechanisms for determining what to store, update, or delete often lead to out-of-date or conflicting information that exacerbates planning errors and hallucinations. Third, the absence of unified schemas for indexing and retrieving heterogeneous data types (text, graphs, audio, video) remains a fundamental architectural gap for LLM-based agents.

Multiagent Interaction

Multiagent LLM systems show promise for solving complex tasks like software development, but still face critical deployment barriers. First, hallucinations and errors from individual agents cascade and amplify throughout the system, undermining the reliability and auditability of the systems. Second, interagent communications cause token flooding and context explosion, incurring substantial computational costs and reducing the efficiency of the systems. Third, existing architectures (sequential or hierarchical) lack principled frameworks for task allocation, role specialization, and conflict resolution, requiring extensive task-specific tuning and manual configuration, which significantly limit the applicability of multiagent LLM systems. Therefore, developing the unified coordination protocols for heterogeneous agent cooperation remains an open challenge for LLM-based agents.

Opportunities for AAMAS Community

Over the past decades, the AAMAS community has developed extensive theories and techniques for trustworthy autonomy, ranging from decision making and knowledge reasoning to principled mechanisms for coordination and cooperation. These foundations offer a rich toolkit for addressing emerging challenges in LLM-based agents due to the intrinsic similarities to the classic problems extensively investigated by the community. In the following sections, we outline several key directions where the expertise of the AAMAS community can make a unique impact in the LLM era.

Symbolic Integration for Verifiable Reasoning

Classical models of planning, decision making, knowledge representation, and automated reasoning can endow LLM-based agents with explicit, verifiable reasoning structures. For example, integrating symbolic and domain-specific solvers to provide accurate feedback signals, enable long-horizon planning, and mitigate hallucinations through constrained generation. Under this paradigm, LLMs function not as monolithic

decision-makers but as interfaces between unstructured natural language and formal symbolic languages, with computationally intensive reasoning delegated to provably correct symbolic systems.

Principled Memory Architectures

The AAMAS community is uniquely positioned to advance memory architectures beyond current approaches. Decades of research on belief revision, knowledge maintenance, case-based reasoning, and multimodal representation learning provide theoretical foundations for agent memory systems. By establishing rigorous criteria for knowledge storage, update mechanisms, and epistemic justification, the community can build memory modules that are dynamically coupled with decision-making processes, grounded in causal relevance, and robust to noise and adversarial perturbations.

Coordination Mechanisms for Multiagent LLM Systems

Classic AAMAS research on coordination, negotiation, and distributed problem solving can contribute to the design of scalable and reliable modern multiagent LLM systems. For instance, established frameworks, like teamwork and coalition structure generation, can inspire architectures that mitigate context overflow, regulate communication bandwidth, and support role adaptation. Similarly, insights from distributed problem solving can help to develop robust coordination protocols for heterogeneous LLM-based agents with different specialized capabilities.

Together, these opportunities highlight the pivotal role the AAMAS community can play in shaping the next generation of AI agents. By combining the generative flexibility of LLMs with principled models, algorithms, and theories developed over decades of AAMAS research, we can move toward agents that are not only more capable but also more trustworthy, robust, and aligned with human and societal values.

REFERENCES

1. "Computer use tool." Claude Developer Platform, 2025. Accessed: Dec. 1, 2025. [Online]. Available: https://platform.claude.com/docs/en/agents-and-tools/tool-use/computer-use-tool
2. G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Commun. ACM*, vol. 54, no. 12, pp. 92–103, 2011, doi: 10.1145/2043174.2043195.
3. T. Brown et al., "Language models are few-shot learners," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 1877–1901.
4. B. G. Buchanan and E. H. Shortliffe, *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA, USA: Addison-Wesley, 1984.
5. "Windsurf editor." Windsurf, 2025. Accessed: Dec. 1, 2025. [Online]. Available: https://windsurf.com/editor
6. S. Wu, "Introducing Devin, the first AI software engineer," *Cognition AI*, Mar. 12, 2024. [Online]. Available: https://cognition.ai/blog/introducing-devin
7. R. H. Crites and A. G. Barto, "Elevator group control using multiple reinforcement learning agents," *Mach. Learn.*, vol. 33, nos. 2–3, pp. 235–262, 1998, doi: 10.1023/A:1007518724497.
8. "Cursor: The best way to code with AI." Cursor, 2025. Accessed: Dec. 1, 2025. [Online]. Available: https://cursor.com/
9. H. B. Enderton, *A Mathematical Introduction to Logic*, 2nd ed. New York, NY, USA: Academic Press, 2001.
10. R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, nos. 3–4, pp. 189–208, 1971, doi: 10.1016/0004-3702(71)90010-5.
11. E. C. Freuder, "In pursuit of the holy grail," *Constraints*, vol. 2, no. 1, pp. 57–61, 1997, doi: 10.1023/A:1009749006768.
12. E. C. Freuder and A. K. Mackworth, "Constraint satisfaction: An emerging paradigm," in *Foundations of Artificial Intelligence*, vol. 2, F. Rossi, P. Van Beek, and T. Walsh, Eds. Amsterdam, The Netherlands: Elsevier, 2006, pp. 13–27.
13. C. Gao et al., "Large language models empowered agent-based modeling and simulation: a survey and perspectives," *Humanities Social Sci. Commun.*, vol. 11, no. 1, 2024, Art. no. 1259, doi: 10.1057/s41599-024-03611-3.
14. P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise, *An Introduction to the Planning Domain Definition Language* (Synthesis Lectures on Artificial Intelligence and Machine Learning). San Rafael, CA, USA: Morgan & Claypool, 2019.
15. H. He et al., "WebVoyager: Building an end-to-end web agent with large multimodal models," 2024, *arXiv:2401.13919*.
16. F.-H. Hsu, *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Princeton, NJ, USA: Princeton Univ. Press, 2022.
17. N. Li, C. Gao, M. Li, Y. Li, and Q. Liao, "EconAgent: Large language model-empowered agents for simulating macroeconomic activities," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2024, pp. 15,523–15,536.
18. T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

19. R. Lowe, Y. I. Wu, A. Tamar, and J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6382–6393.
20. C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, "The AI scientist: Towards fully automated open-ended scientific discovery," 2024, *arXiv:2408.06292*.
21. J. Luo et al., "Large language model agent: A survey on methodology, applications and challenges," 2025, *arXiv:2503.21460*.
22. R. Lupoiu, Y. Shao, T. Dai, C. Mao, K. Edée, and J. A. Fan, "A multi-agentic framework for real-time, autonomous freeform metasurface design," *Sci. Adv.*, vol. 11, no. 44, 2025, Art. no. eadx8006, doi: 10.1126/sciadv.adx8006.
23. A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Augmenting large language models with chemistry tools," *Nature Mach. Intell.*, vol. 6, no. 5, pp. 525–535, 2024, doi: 10.1038/s42256-024-00832-8.
24. V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.
25. "Specification." Model Context Protocol, 2025. Accessed: Dec. 1, 2025. [Online]. Available: https://mcp.mintlify.app/specification/2025-03-26/index
26. A. Newell, J. C. Shaw, and H. A. Simon, "Report on a general problem solving program," in *Proc. Int. Conf. Inf. Process.*, 1959, pp. 256–264.
27. A. Newell and H. A. Simon, *Human Problem Solving*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1972.
28. A. Y. Ng et al., "Autonomous inverted helicopter flight via reinforcement learning," in *Proc. 9th Int. Symp. Exp. Robot.*, 2006, pp. 363–372.
29. A. Novikov et al., "AlphaEvolve: A coding agent for scientific and algorithmic discovery," 2025, *arXiv:2506.13131*.
30. "Computer-using agent," *OpenAI*, Jan. 23, 2025. [Online]. Available: https://openai.com/index/computer-using-agent/
31. L. Ouyang et al., "Training language models to follow instructions with human feedback," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2022, pp. 27,730–27,744.
32. 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 *Proc. 36th Annu. ACM Symp. User Interface Softw. Technol. (UIST)*, 2023, pp. 1–22.
33. S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive APIs," in *Proc. 38th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2024, pp. 126,544–126,565.
34. A. Plaat, M. van Duijn, N. van Stein, M. Preuss, P. van der Putten, and K. J. Batenburg, "Agentic large language models, a survey," *J. Artif. Intell. Res.*, vol. 84, pp. 29:1–29:74, Dec. 2025, doi: 10.1613/jair.1.18675.
35. C. Qian et al., "ChatDev: Communicative agents for software development," in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (Vol. 1: Long Papers)*, 2024, pp. 15,174–15,186.
36. O. Ram et al., "In-context retrieval-augmented language models," in *Transactions of the Association for Computational Linguistics*, vol. 11, Asli Celikyilmaz, Ed. Cambridge, MA, USA: MIT Press, 2023, pp. 1316–1331, doi: 10.1162/tacl_a_00605.
37. T. Rashid, M. Samvelyan, C. Schroeder De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 178, pp. 1–51, 2020.
38. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.
39. M. H. Samaei, F. D. Sahneh, L. W. Cohnstaedt, and C. M. Scoglio, "EpidemIQs: Prompt-to-paper LLM agents for epidemic modeling and analysis," *IEEE Trans. Artif. Intell.*, early access, 2025, doi: 10.1109/TAI.2026.3666830.
40. T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2023, pp. 68,539–68,551.
41. J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
42. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
43. N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2023, pp. 8634–8652.
44. D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, doi: 10.1038/nature16961.
45. D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017, doi: 10.1038/nature24270.
46. P. Stone, R. S. Sutton, and G. Kuhlmann, "Reinforcement learning for RoboCup Soccer keepaway," *Adaptive Behav.*, vol. 13, no. 3, pp. 165–188, 2005, doi: 10.1177/105971230501300301.
47. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

48. W. Tan et al., "Cradle: Empowering foundation agents towards general computer control," in *Proc. 42nd Int. Conf. Mach. Learn.*, PMLR, 2025, pp. 58,658–58,725.
49. G. Tesauro, "Temporal difference learning and TD-Gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995, doi: 10.1145/203330.203343.
50. G. Tom et al., "Self-driving laboratories for chemistry and materials science," *Chem. Rev.*, vol. 124, no. 16, pp. 9633–9732, 2024, doi: 10.1021/acs.chemrev.4c00055.
51. X. Wang et al., "Self-consistency improves chain of thought reasoning in language models," 2022, *arXiv:2203.11171*.
52. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2022, pp. 24,824–24,837.
53. J. Wei et al., "From AI for science to agentic science: A survey on autonomous scientific discovery," 2025, *arXiv:2508.14111*.
54. A. Yang et al., "Qwen3 technical report," 2025, *arXiv:2505.09388*.
55. J. Yang et al., "SWE-agent: Agent-computer interfaces enable automated software engineering," in *Proc. 38th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2024, pp. 50,528–50,652.
56. S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2023, pp. 11,809–11,822.
57. S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," 2023, *arXiv:2210.03629*.
58. Z. Zhang et al., "A survey on the memory mechanism of large language model-based agents," *ACM Trans. Inf. Syst.*, vol. 43, no. 6, pp. 1–47, 2025, doi: 10.1145/3748302.

BO AN is a President's Chair professor at Nanyang Technological University, 6397987, Singapore. His research interests include artificial intelligence, multiagent systems, computational game theory, and reinforcement learning. An received his Ph.D. degree in computer science from the University of Massachusetts, Amherst. He is a fellow of the Association for the Advancement of Artificial Intelligence. Contact him at boan@ntu.edu.sg.