Autonomous Agentic Reasoning and Tool Integration in LLM Systems: A Survey

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly being deployed as autonomous agents that interact with external tools and APIs to accomplish complex goals. This requires two core abilities working together: the capacity to reason and plan, and the ability to perform function calling. Although both capabilities have been studied separately, their combination in agent systems creates new challenges that make a unified treatment timely and necessary.

In this survey, we review how reasoning and function calling interact in LLM-based agent systems. We examine how structured reasoning improves tool selection and parameter generation, and how tool access in turn strengthens reasoning by offloading computation and grounding intermediate steps. We cover training methodologies spanning prompt engineering, supervised fine-tuning, and reinforcement learning, as well as orchestration patterns from single-agent single-turn interactions to multi-agent and multi-modal settings. We also review evaluation benchmarks and applications spanning scientific research, healthcare, finance, software engineering, and robotics. Finally, we outline the open challenges and future directions.

1 Introduction

Large language models (LLMs) have emerged as powerful tools with wide-ranging applications in challenging domains such as programming, reasoning, and multi-modal interaction (Wei et al., 2022; Yang et al., 2023b; Gemini Team, 2023). As these models are increasingly deployed to address real-world problems, there is growing demand for them to operate as autonomous agents that can make intelligent decisions within specific environments (Wang et al., 2024; Mialon et al., 2023). This transition to autonomous agency requires two core competencies: (a) the ability to reason and plan (Wei et al., 2022; Yao et al., 2023a,b), and (b) the ability to perform function calling: that is, to identify, execute, and interact with external tools and APIs (Schick et al., 2023; Patil et al., 2023; Qin et al., 2023b).

While these capabilities have been studied independently, their integration in agent systems presents unique challenges and opportunities (Qin et al., 2023a,a). Effective function calling requires sophisticated reasoning about when to use tools, which tools to select, how to generate appropriate parameters, and how to interpret results (Dang et al., 2025; Paranjape et al., 2023; Ruan et al., 2023). Conversely, function calling extends reasoning capabilities by enabling agents to access external knowledge, perform computations, and interact with environments (Schick et al., 2023; Gou et al., 2023; Lu et al., 2023).

In this survey, we provide a comprehensive and structured review of how reasoning and function calling capabilities combine in LLM-based agent systems. We trace the development of reasoning paradigms, from chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2022) to tree-based search (Yao et al., 2023a; Besta et al., 2024) and planning architectures (Song et al., 2023; Wang et al., 2023), and examine how these paradigms support effective tool use (Yao et al., 2023b; Gao et al., 2023; Chen et al., 2022). We then turn to the mechanisms by which models learn to call functions reliably, covering prompting-based approaches (Dang et al., 2025; Paranjape et al., 2023), supervised fine-tuning (Schick et al., 2023; Tang et al., 2023), reinforcement learning (Schulman et al., 2017; Hao et al., 2025), and parameter-efficient techniques (Hu et al., 2022; Dettmers et al., 2023). Building on these foundations, we analyze how reasoning and tool orchestration interact at multiple levels of complexity, from single-agent interactions (Schick et al., 2023; Qin et al., 2023b) to multi-agent collaborative systems

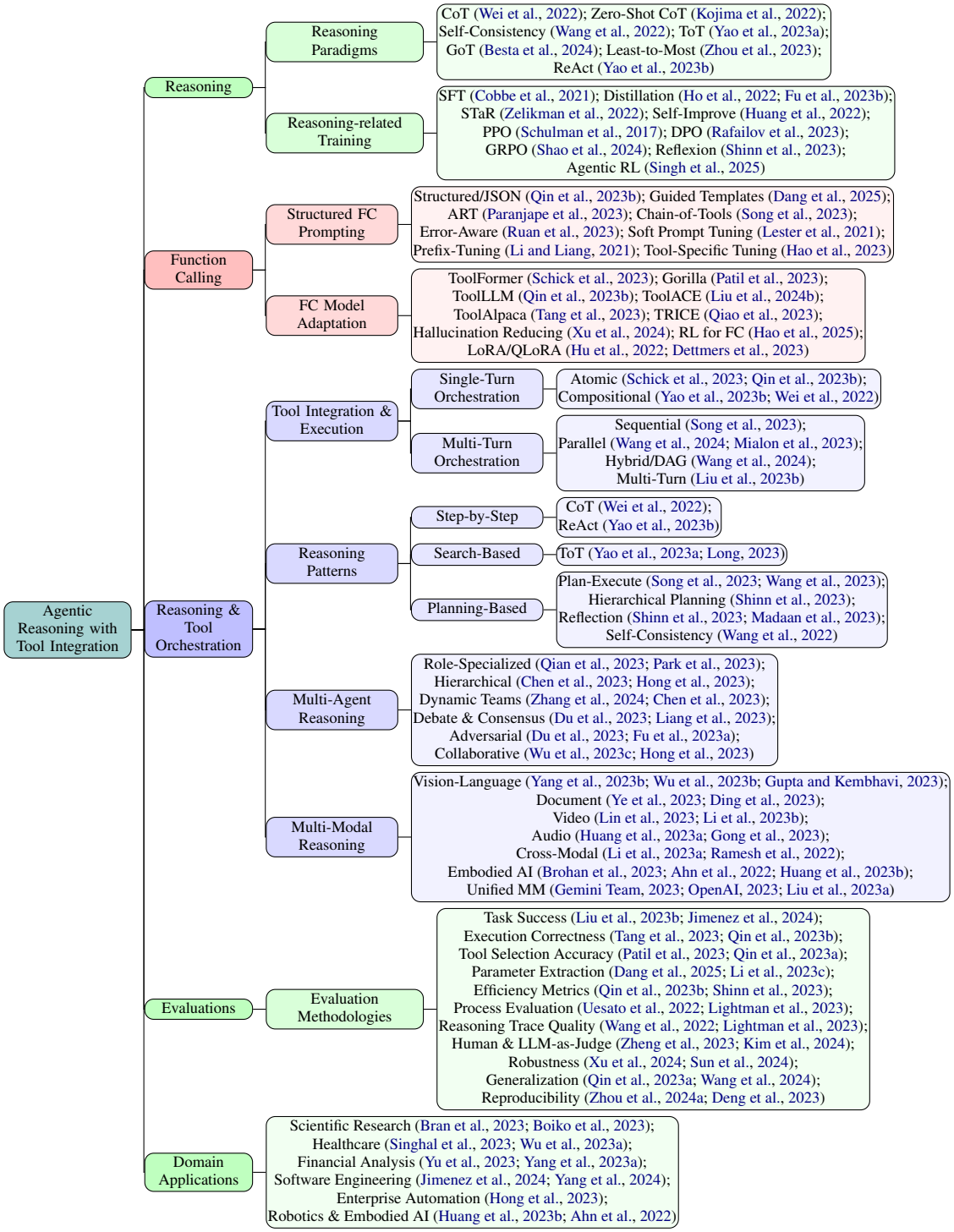

Figure 1: Taxonomy of agentic reasoning and tool integration in LLM systems. The tree organizes the survey along six dimensions: reasoning paradigms and training methods, function calling techniques, orchestration patterns (including multi-agent and multi-modal reasoning), evaluation methodologies, and domain applications.

(Wu et al., 2023c; Hong et al., 2023; Park et al., 2023) and multi-modal environments (Yang et al., 2023b; Brohan et al., 2023; Ahn et al., 2022). We further survey evaluation benchmarks (Liu et al., 2023b; Jimenez et al., 2024; Zheng et al., 2023) and highlight domain applications including scientific research (Bran et al., 2023; Boiko et al., 2023), healthcare (Singhal et al., 2023; Moor et al., 2023), finance (Yu et al., 2023; Yang et al., 2023a), software engineering (Jimenez et al., 2024; Yang et al., 2024), and robotics (Huang et al., 2023b; Ahn et al., 2022). Finally, we identify key open challenges and outline promising directions for future research. Figure 2 provides an overview.

2 Background

2.1 Large Language Models as Agents

Large language models have evolved from passive question-answering systems to autonomous agents capable of goal-directed behavior (Wang et al., 2024; Mialon et al., 2023). An LLM agent typically consists of four core components: (1) *Perception*: processing observations from the environment, including user queries and function outputs; (2) *Reasoning*: planning actions and making decisions based on accumulated observations (Yao et al., 2023b); (3) *Action*: executing function calls or manipulating the environment to achieve specified goals (Schick et al., 2023); and (4) *Memory*: maintaining context across interactions for coherent multi-turn engagement (Park et al., 2023). This agent paradigm shifts LLMs from single-turn response generation to interactive, multi-step problem solving that requires sustained reasoning, adaptive planning, and iterative refinement.

2.2 Function Calling Fundamentals

Function calling enables LLMs to interact with external tools and APIs by generating structured outputs that specify which functions to invoke and with what parameters (Schick et al., 2023; Patil et al., 2023). The core mechanism involves parsing user requests, identifying relevant tools, extracting appropriate parameters, and formatting these into valid function calls, which typically represented as JSON objects (Qin et al., 2023b; Dang et al., 2025). Despite advances in model capabilities, function calling remains challenging due to ambiguous user intent, complex parameter dependencies, and incomplete tool documentation (Ruan et al., 2023; Tam et al., 2024).

Many proprietary models now provide native function calling support, while open-source models typically require fine-tuning or prompt engineering to acquire this capability (Tang et al., 2023; Liu et al., 2024b). Approaches range from full retraining to parameter-efficient methods such as LoRA (Hu et al., 2022) and prompt-based alternatives including few-shot prompting with tool descriptions (Patil et al., 2023; Qin et al., 2023b); we discuss these in detail in §6.

2.3 Overview of Reasoning in Large Language Models

Reasoning in LLMs refers to the capacity to derive conclusions through structured intermediate steps rather than producing direct outputs (Wei et al., 2022). Formally, let $\mathcal{M}$ denote a language model parameterized by θ, with input x and output y. Standard next-token prediction factorizes as:

$$P_\theta(y \mid x) = \prod_{t=1}^{T} P_\theta(y_t \mid x, y_{<t}). \tag{1}$$

Reasoning-augmented generation extends this by introducing a latent chain of thought $z = (z_1, \ldots, z_K)$, such that:

$$P_\theta(y \mid x) = \sum_{z \in \mathcal{Z}} P_\theta(z \mid x)\, P_\theta(y \mid x, z), \tag{2}$$

where $\mathcal{Z}$ is the space of all possible reasoning chains. This idea, introduced through Chain-of-Thought (CoT) prompting (Wei et al., 2022), significantly improves performance on multi-step arithmetic, commonsense reasoning, and symbolic tasks.

Beyond single-chain inference, methods such as Tree-of-Thought (ToT) (Yao et al., 2023a) and Graph-of-Thought (GoT) (Besta et al., 2024) model reasoning as a search over a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, selecting the optimal path z^* via:

$$z^* = \arg\max_{z \in \mathcal{T}} \sum_{k=1}^{|z|} r(z_k), \tag{3}$$

where $r(z_k)$ is a value function assigned to each reasoning step. Reinforcement-learning approaches (Schulman et al., 2017; Shao et al., 2024; Singh et al., 2025) further optimize the model by maximizing expected task reward while regularizing against a reference policy, capturing the full progression from flat token prediction to structured, reward-guided reasoning.

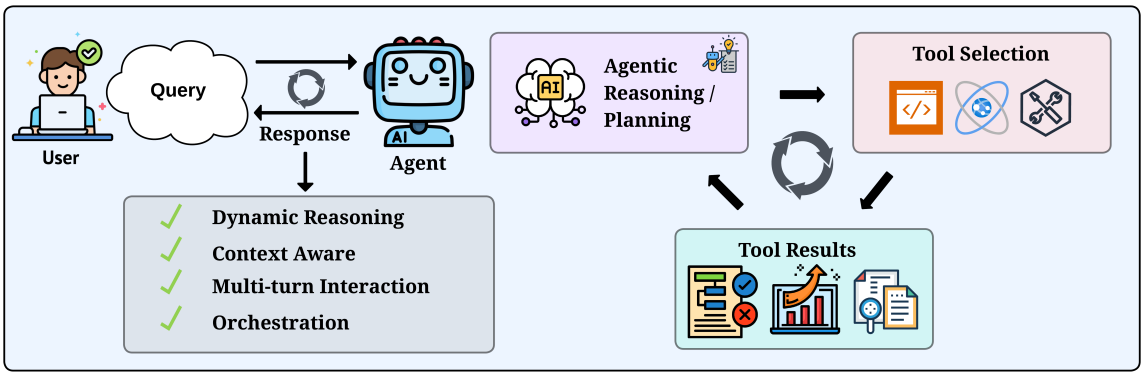

Figure 2: Overview of agentic reasoning and tool integration in LLM Systems. This figure illustrates the interaction between a user and an AI agent, where the agent performs agentic reasoning and planning to decompose tasks, select appropriate tools, executes them, and incorporates the tool results back into its reasoning cycle. The agent enhance its capabilities through dynamic reasoning, context awareness, multi-turn interaction, and orchestration etc.

3 Reasoning Strategies for Function Calling

A key challenge in agentic LLM systems is deciding *when* to call a function and *which* function to use. Common errors, such as wrong parameter values, choosing the wrong tool, and misunderstanding what the user wants, often happen because the model does not fully understand the user's goal or how the tool works (Dang et al., 2025).

These problems are made worse by the technical demands of function calling itself. Tam et al. (2024) showed that simply requiring JSON-formatted outputs dropped accuracy by more than 27 percentage points on GSM8K, and Sun et al. (2024) found that asking a model to understand a query, pick the right tool, and format the output all at once can cause more than 20% accuracy loss in some models. Although Chain-of-Thought (CoT) prompting helps with general reasoning, this kind of free-form thinking is often not enough for function calling tasks that need structured outputs (Dang et al., 2025). Paranjape et al. (2023) showed this through ART, showing that unstructured CoT breaks down when tool calls must be closely tied to the reasoning steps, while a structured approach that pauses at each tool call performs much better than both few-shot and automatic CoT methods. Building on this, Liu et al. (2024b) found that dealing with parameter types, constraints, and dependencies requires step-by-step reasoning that free-form CoT regularly fails to provide. Such finding shows that the model needs clear reasoning guidance at each stage of the function calling process, rather than only at the final output.

To address these issues, Dang et al. (2025) propose a structured reasoning framework that guides the model through function call generation step by step, encouraging it to reason carefully before executing a call and thereby improving both accuracy and interpretability. Other approaches address the problem from different angles. TRICE (Qiao et al., 2023) uses post-execution feedback to help models learn from mistakes, while Xu et al. (2024) focus on improving tool selection to reduce made-up API calls.

In summary, structured reasoning guidance at each stage of the function calling process, rather than free-form thinking, is key to reliable tool use.

4 Function Calls for Reasoning

Tools fundamentally strengthen LLM reasoning by preventing reasoning chains from breaking down at execution steps. While a model's reasoning may correctly identify the sequence of operations needed, tools ensure each step is executed accurately. This is directly evidenced by Program-Aided Language Models (PAL) (Gao et al., 2023), which delegate computation to a Python interpreter rather than relying on the model's internal arithmetic, and by the Program of Thoughts framework (Chen et al., 2022), which explicitly disentangles reasoning from computation, preventing execution-level failures from corrupting otherwise valid reasoning chains.

This creates a powerful synergy, where reasoning provides problem decomposition and planning, while tools provide reliable execution. The REACT framework (Yao et al., 2023b) demonstrates this by interleaving chain-of-thought reasoning with tool-action steps, consistently outperforming both reasoning-only and tool-only baselines on

knowledge-intensive tasks. Without tools, even valid reasoning chains can fail due to arithmetic errors, outdated information, or hallucinated facts.

Beyond correcting execution failures, tool access improves reasoning quality itself. When models can offload computation to calculators or code interpreters, they reason at a higher level of abstraction, focusing on problem structure rather than computational details. TOOLFORMER (Schick et al., 2023) showed that models can autonomously learn to invoke tools, while TOOLLLM (Qin et al., 2023b) demonstrated better generalization across thousands of real-world APIs. Composing heterogeneous tools further enables solving tasks that neither the base model nor any single tool could handle alone (Lu et al., 2023; Qin et al., 2023a).

5 Reasoning and Tool Orchestration

Reasoning patterns define *how* agents think and make decisions, while tool orchestration patterns define *how* agents interact with external systems and manage workflows (Wang et al., 2024; Ke et al., 2025; Singh et al., 2025). These two dimensions are complementary: orchestration concerns the structural arrangement of tool calls, while reasoning governs the strategy used to select, sequence, and interpret them. We organize orchestration along interaction scope (single-turn vs. multi-turn) and execution topology (sequential, parallel, or hybrid), and reasoning into step-by-step, search-based, and planning-based strategies.

5.1 Tool Integration and Execution Patterns

In single-turn orchestration, the model calls one or more tools within a single reasoning step and synthesizes results without persisting state across turns (Schick et al., 2023; Qin et al., 2023b). This covers both atomic interactions, where a single tool handles a well-defined query, and compositional interactions, where multiple tools are chained to answer more complex questions (Lu et al., 2023; Paranjape et al., 2023). Multi-turn orchestration extends this across a conversation, where context accumulates and the agent must track earlier tool calls, decide when to reuse cached results versus issuing fresh calls, and manage a growing context window (Liu et al., 2023b, 2024a; Ruan et al., 2023).

Regarding execution topology, sequential execution runs tools in strict dependency order, where each output feeds directly into the next step, though slow or expensive calls can stall the entire pipeline (Song et al., 2023; Wang et al., 2023). Parallel execution runs independent subtasks concurrently to reduce overall latency, requiring the agent to identify safe concurrent calls and handle asynchronous results and partial failures gracefully (Wang et al., 2024; Mialon et al., 2023; Qin et al., 2023b). Most complex workflows use hybrid execution, combining both strategies by constructing a DAG of tool dependencies to determine an efficient execution schedule (Lu et al., 2023; Ruan et al., 2023; Wu et al., 2023c).

5.2 Reasoning Patterns for Tool-Augmented Agents

Step-by-step reasoning methods make intermediate thinking explicit before each tool call. CoT prompting (Wei et al., 2022; Kojima et al., 2022) improves tool selection and parameter extraction on multi-step tasks, while ReAct (Yao et al., 2023b) extends this into interactive settings by interleaving reasoning and acting: the agent produces a thought, executes a tool call, observes the result, and reasons again, enabling dynamic adaptation based on what each tool returns (Singh et al., 2025; Gou et al., 2023).

Search-based methods treat problem-solving as a search over possible reasoning paths rather than committing to a single chain of thought. Tree of Thoughts (ToT) (Yao et al., 2023a; Long, 2023) lets agents explore multiple reasoning branches simultaneously and backtrack when a branch proves unfruitful, which is particularly valuable when several valid tool-use strategies exist. Graph of Thoughts (GoT) (Besta et al., 2024) generalizes this further by allowing arbitrary graph-structured reasoning, enabling the agent to merge and refine thoughts across branches rather than treating them as independent paths.

Planning-based approaches separate strategy formulation from execution, reasoning over longer horizons before any tools are called. Plan-and-Execute architectures (Song et al., 2023; Wang et al., 2023) first produce a high-level tool sequence, then carry it out while adapting when unexpected results arise, making the decision process easier to inspect and replan. Hierarchical planning extends this by recursively decomposing high-level goals into lower-level subtasks, each of which may itself require a sequence of tool calls (Wang et al., 2024; Ruan et al., 2023), allowing the agent to reason at multiple levels of abstraction. Finally, reflection-

based methods let agents evaluate and revise their own tool usage after execution (Shinn et al., 2023). Self-consistency (Wang et al., 2022) improves reliability by generating multiple independent reasoning chains and selecting the most frequent answer, while the Reflexion framework stores verbal self-reflections in persistent memory, enabling agents to learn from past mistakes across episodes without gradient-based updates (Shinn et al., 2023; Gou et al., 2023).

5.3 Multi-Agent Reasoning with Tool Integration

Multi-agent systems extend single-agent capabilities through collaboration, specialization, and division of labor. In role-specialized architectures, planners, executors, critics, and verifiers form robust pipelines (Qian et al., 2023; Park et al., 2023), while hierarchical coordination enables manager-worker patterns that scale across complex tool ecosystems (Chen et al., 2023; Hong et al., 2023). Dynamic team formation allows agents to self-organize based on task requirements (Zhang et al., 2024), with shared memory and communication protocols preventing redundant API calls and keeping information flowing efficiently (Park et al., 2023; Zhou et al., 2024b).

Beyond structure, multi-agent systems derive power from diverse interaction strategies. Collaborative agents share intermediate tool results to reach goals beyond any single agent (Wu et al., 2023c; Hong et al., 2023), debate mechanisms improve reliability through structured critique of competing solutions (Du et al., 2023; Liang et al., 2023), and adversarial setups surface edge cases through red-teaming (Du et al., 2023; Fu et al., 2023a).

5.4 Multi-Modal Reasoning with Tool Integration

Multi-modal reasoning extends the orchestration patterns above beyond text by coordinating modality-specific tools across vision, documents, video, audio, and embodied control (Yang et al., 2023b; Huang et al., 2023a; Brohan et al., 2023; Ahn et al., 2022). Unified architectures treat different modalities as interchangeable token streams, with tools providing modality-specific transformations as needed (Gemini Team, 2023; OpenAI, 2023; Liu et al., 2023a). We discuss domain-specific applications in §8.

6 Training and Optimization

6.1 Training Methodologies

Developing reasoning and tool-use capabilities requires training methodologies that go beyond standard language modeling. Three primary approaches have emerged: prompt engineering, supervised fine-tuning, and reinforcement learning, each offering distinct tradeoffs between cost, capability, and generalization.

6.1.1 Prompt Engineering and Prompt Tuning

Prompt engineering enhances reasoning and tool-use without modifying model parameters. For reasoning, CoT prompting (Wei et al., 2022) encourages step-by-step decomposition, while zero-shot variants (Kojima et al., 2022), Self-Consistency (Wang et al., 2022), and ToT (Yao et al., 2023a) extend this to multi-path strategies. For tool use, ReAct (Yao et al., 2023b) interleaves reasoning with tool actions, and structured JSON/XML formats (Patil et al., 2023; Qin et al., 2023b) enable reliable function call generation. Prompt tuning offers a parameter-efficient alternative by optimizing continuous embeddings while keeping weights frozen (Lester et al., 2021; Li and Liang, 2021), though it risks overfitting to specific reasoning patterns or tool sets.

6.1.2 Supervised Fine-Tuning

Supervised fine-tuning (SFT) trains models directly on reasoning exemplars and tool-use demonstrations. Given a dataset $\mathcal{D} = \{(x_i, z_i, y_i)\}$ of input, reasoning chain, and output triples, SFT minimizes the negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,z,y)\sim\mathcal{D}} \left[\log P_\theta(z, y \mid x)\right], \quad (4)$$

where jointly modeling the reasoning chain z alongside the output y teaches models to produce coherent intermediate steps rather than direct answers (Cobbe et al., 2021; Wei et al., 2022). For tool use, the same objective applies with z representing a function call sequence, teaching models to map natural language intents to valid API invocations (Schick et al., 2023; Patil et al., 2023).

Dataset construction critically shapes SFT outcomes. Distillation uses stronger models to generate reasoning traces for weaker ones (Ho et al., 2022; Fu et al., 2023b), while self-training methods such as STaR (Zelikman et al., 2022) have models generate and filter their own chains iteratively. For tool use, synthetic data generation covers rare tools

and parameter combinations at scale (Tang et al., 2023; Lu et al., 2023). Parameter-efficient methods such as LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023) enable rapid adaptation without full retraining, and execution-aware training incorporates actual API results to teach models which function calls succeed or fail (Ruan et al., 2023; Qiao et al., 2023).

6.1.3 Reinforcement Learning Fine-Tuning

RL fine-tuning optimizes reasoning and tool use by directly maximizing task success through trial and error, rather than mimicking demonstrations. The general RL objective is:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, z\sim P_\theta(\cdot|x)}\left[R(x,z,y)\right] - \lambda\,\mathrm{KL}(P_\theta \,\|\, P_{\theta_0}) \tag{5}$$

where $R(x, z, y)$ is a task-level reward over the reasoning chain z and output y, and the KL term regularizes against a reference policy P_{θ_0} to prevent reward hacking. PPO (Schulman et al., 2017) is widely adopted for its training stability, though simpler alternatives have since emerged.

Direct Preference Optimization (DPO) (Rafailov et al., 2023; Ge et al., 2026) eliminates explicit reward modeling by directly optimizing over preference pairs (z^+, z^-) of winning and losing reasoning or tool-use trajectories, increasing the log-likelihood ratio of z^+ over z^- relative to a reference policy P_{θ_0} (full derivation in Appendix A). Group Relative Policy Optimization (GRPO) (Shao et al., 2024) extends this by comparing groups of trajectories simultaneously, capturing efficiency versus reliability tradeoffs across multiple candidate paths.

Reward Design. Reward signals are central to RL fine-tuning. Outcome-based rewards provide binary success signals from verifiable task completion (Lightman et al., 2023), while process-based rewards give dense step-level feedback on intermediate reasoning steps or individual tool calls (Uesato et al., 2022). Efficiency penalties discourage redundant tool calls or reasoning steps (Qin et al., 2023b), and safety constraints penalize unauthorized API invocations or policy violations. Advanced techniques such as verifier-guided RL (Cobbe et al., 2021), execution-based RL (Gou et al., 2023), and hierarchical RL (Song et al., 2023) further improve robustness and generalization to novel tool combinations and long-horizon reasoning tasks.

7 Evaluation Methodologies

Evaluating reasoning with function calls is more challenging than evaluating single-shot QA because success depends not only on the final response but also on how the agent changes the environment over multiple steps. Multi-turn and parallel execution patterns (§5) require tracking cumulative state changes across an entire conversation. We organize evaluation into three complementary perspectives: (1) *state-based correctness and task success*, (2) *process evaluation*, and (3) *robustness, generalization, and reproducibility*.

7.1 State-Based Evaluation

For multi-step tool calling, state-based evaluation has emerged as a primary paradigm. In realistic environments, a task may be completed through many valid action sequences: agents may ask clarifying questions, recover from failed calls, or use different tool orderings while still reaching the correct outcome. This makes trajectory matching against a single reference path brittle, especially as the number of valid trajectories grows combinatorially with retries, optional information gathering, and alternative task decompositions. Enumerating all such correct paths is therefore intractable in realistic settings.

Recent benchmarks address this challenge through state-based evaluation. AppWorld evaluates whether the final database state matches a set of valid gold states rather than a single reference trajectory (Trivedi et al., 2024). Similarly, τ-bench compares the final database state against an annotated goal state and reports pass^k to measure consistency across runs (Yao et al., 2025). BFCL has likewise moved beyond single-call accuracy to stateful multi-step agentic evaluation, reflecting the need to assess memory, long-horizon interaction, and dynamic decision making (Patil et al., 2025; Mao et al., 2024). By focusing on the resulting environment state rather than one canonical trace, these approaches better capture whether the agent satisfied the user goal while avoiding unwanted side effects such as incorrect updates or deletions (Trivedi et al., 2024; Yao et al., 2025; Mao et al., 2024). A remaining challenge is scaling state-based evaluation to settings where gold-state annotation is expensive. Chuang et al. (2026) propose proxy state-based evaluation, which uses lightweight proxy states derived from tool-call side effects to approximate full environment verifica-

tion, enabling scalable and verifiable reward signals for multi-turn tool-calling agents.

7.2 Process Evaluation and Failure Diagnosis

State-based evaluation alone is not sufficient. Some tool calls are read-only and leave little persistent trace, and agents may reach the correct end state through inefficient or unsafe behavior. Process-oriented metrics therefore remain important complements. BFCL explicitly combines state-based and response-based evaluation in multi-turn settings, using state checks for state-changing operations and response-based checks when state alone is insufficient (Mao et al., 2024).

Useful diagnostic metrics include execution correctness, tool selection accuracy, parameter extraction precision, reasoning-trace quality, and efficiency measured by tool-call count, latency, and token cost (Tang et al., 2023; Qin et al., 2023b; Patil et al., 2023; Qin et al., 2023a; Dang et al., 2025). Human evaluation remains important for subjective qualities such as helpfulness and safety (Zheng et al., 2023; Kim et al., 2024), while LLM-as-judge methods offer a scalable proxy.

7.3 Robustness, Generalization, and Reproducibility

A complete evaluation framework must also assess reliability under perturbations and across repeated trials. Robustness benchmarks test resilience to adversarial prompts, malformed tool outputs, and API failures (Xu et al., 2024; Sun et al., 2024). Generalization benchmarks examine transfer to unseen tools, novel tool compositions, and longer-horizon tasks (Qin et al., 2023a; Wang et al., 2024). Reproducibility remains challenging because performance may depend on live APIs, stochastic sampling, and changing environments (Zhou et al., 2024a).

Taken together, recent benchmarks suggest a layered evaluation framework for tool-augmented LLM agents: state-based task success as the primary criterion for multi-step interaction, process-based metrics for diagnosis, and robustness and reproducibility measures for deployment readiness.

8 Domain Applications

Reasoning-capable LLMs integrated with domain-specific tools have shown transformative potential across two broad settings. In knowledge-intensive domains, agents retrieve, synthesize, and reason over specialized information: accelerating scientific discovery (Bran et al., 2023; Boiko et al., 2023), analyzing medical literature and supporting treatment planning (Singhal et al., 2023; Wu et al., 2023a), and orchestrating market data APIs for financial analysis (Yu et al., 2023; Yang et al., 2023a). In action-oriented domains, agents directly manipulate external systems: autonomously resolving GitHub issues in software engineering (Jimenez et al., 2024; Yang et al., 2024), orchestrating enterprise workflows across CRM and communication platforms (Hong et al., 2023), and planning physical tasks in robotics and embodied AI (Huang et al., 2023b; Ahn et al., 2022). Together, these applications demonstrate that combining robust reasoning with domain-specific tool integration enables agents to tackle tasks requiring both deep knowledge and coordinated multi-step action.

9 Challenges and Future Directions

Despite significant progress, key challenges remain. Agents are prone to reasoning failures that cascade through multi-step tool chains, where a single incorrect tool selection or malformed parameter can derail an entire trajectory (Valmeekam et al., 2023; Gou et al., 2023), and context length constraints limit the number of steps an agent can sustain (Liu et al., 2024a). Tool-augmented agents also introduce new attack surfaces, including adversarial prompts that manipulate tool selection, unauthorized API invocations, and hallucinated calls to non-existent endpoints (Xu et al., 2024; Sun et al., 2024). On the evaluation side, existing benchmarks cover different aspects, single-turn accuracy, multi-turn state correctness, or process quality, but no unified framework spans all dimensions, and reproducibility is complicated by dependence on live APIs and stochastic sampling (Liu et al., 2023b; Jimenez et al., 2024; Zhou et al., 2024a).

Future work should focus on adaptive tool selection that dynamically discovers and composes unfamiliar tools at inference time (Qin et al., 2023b; Hao et al., 2023), explainability methods that make tool-use decisions interpretable (Zhao et al., 2024; Xu et al., 2024), human-agent collaboration where users can inspect and guide agent tool choices (Wu et al., 2023c), and standardized evaluation frameworks that enable fair cross-method comparison (Liu et al., 2023b; Xi et al., 2024). Progress across these dimensions will be essential to realizing the full potential of tool-augmented reasoning agents.

Limitations

This survey has several limitations. First, our coverage is predominantly focused on text-based LLM agents. Although we discuss multi-modal extensions (e.g., vision, audio, robotics), we do not deeply analyze how reasoning and function calling differ when perception itself is multi-modal, such as in embodied agents that must ground tool use in visual observations.

Second, the field is evolving rapidly, and many proprietary systems (e.g., GPT-4, Claude, Gemini) do not publish detailed descriptions of their function calling internals. Our survey therefore skews toward methods with publicly available technical details, which may underrepresent the design choices and capabilities of closed-source models.

Third, this survey does not provide direct empirical comparisons between the methods discussed. While we reference reported results from individual papers, differences in evaluation setups, benchmarks, and model versions make cross-paper comparisons difficult. A standardized re-evaluation of representative methods would strengthen the conclusions drawn here.

Fourth, we do not systematically address practical deployment considerations such as latency, cost, and rate limits of external API calls, which are critical factors when scaling tool-augmented agents to production settings.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and 1 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, and 1 others. 2023. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. *arXiv preprint arXiv:2308.10848*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yun-Shiuan Chuang, Chaitanya Kulkarni, Alec Chiu, Avinash Thangali, Zijie Pan, Shivani Shekhar, Yirou Ge, Yixi Li, Uma Kona, Linsey Pang, and 1 others. 2026. Toward scalable verifiable reward: Proxy state-based evaluation for multi-turn tool-calling llm agents. *arXiv preprint arXiv:2602.16246*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Hy Dang, Tianyi Liu, Zhuofeng Wu, Jingfeng Yang, Haoming Jiang, Tao Yang, Pei Chen, Zhengyang Wang, Helen Wang, Huasheng Li, Bing Yin, and Meng Jiang. 2025. Improving large language models function calling and interpretability via guided-structured templates. In *Proceedings of the 2025*

Conference on Empirical Methods in Natural Language Processing, pages 24426–24442, Suzhou, China. Association for Computational Linguistics.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Fangqiang Ding, Andras Palffy, Dariu M. Gavrila, and Chris Xiaoxuan Lu. 2023. Hidden gems: 4d radar scene flow learning using cross-modal supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9340–9349.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023a. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023b. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *International Conference on Machine Learning*.

Yirou Ge, Yixi Li, Alec Chiu, Shivani Shekhar, Zijie Pan, Avinash Thangali, Yun-Shiuan Chuang, Chaitanya Kulkarni, Uma Kona, Linsey Pang, and 1 others. 2026. Ada-rs: Adaptive rejection sampling for selective thinking. *arXiv preprint arXiv:2602.19519*.

Gemini Team. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Yuan Gong, Hongyin Luo, Alexander H. Liu, Leonid Karlinsky, and James Glass. 2023. Listen, think, and understand. *arXiv preprint arXiv:2305.10790*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962.

Bingguang Hao, Zengzhuang Xu, Maolin Wang, Yuntao Wen, Yicheng Chen, Cunyin Peng, Long Chen, Dong Wang, Xiangyu Zhao, Jinjie Gu, Chenyi Zhuang, and Ji Zhang. 2025. Reasoning through exploration: A reinforcement learning framework for robust function calling. *arXiv preprint arXiv:2508.05118*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in Neural Information Processing Systems*, 36.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, and 1 others. 2023a. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. 2023b. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, and 1 others. 2025. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and

Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023c. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024b. ToolACE: Winning the points of LLM function calling. *arXiv preprint arXiv:2409.00920*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*.

Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Jason Huang, Vishnu Suresh, Yixin Huang, Xiaowen Yu, Joseph E. Gonzalez, and Shishir G. Patil. 2024. Bfcl v3: Multi-turn & multi-step function calling evaluation. `https://gorilla.cs.berkeley.edu/blogs/13_bfcl_v3_multi_turn.html`. Official Gorilla blog post, release date 2024-09-19, last updated 2024-12-10.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: A survey. *arXiv preprint arXiv:2302.07842*.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Cyril Zakka, Yash Dalmia, Eduardo Pontes Reis, Pranav Rajpurkar, and Jure Leskovec. 2023. Med-flamingo: a multimodal medical few-shot learner. *arXiv preprint arXiv:2307.15189*.

OpenAI. 2023. Gpt-4v(ision) system card. *OpenAI Technical Report*.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Shuofei Qiao, Honghao Gui, Chengfei Lv, Qianghuai Jia, Huajun Chen, and Ningyu Zhang. 2023. Making language models better tool learners with execution feedback. *arXiv preprint arXiv:2305.13068*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, and 1 others. 2023a. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023b. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. *arXiv preprint arXiv:2307.16789*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Task planning and tool usage of large language model-based ai agents. *arXiv preprint arXiv:2308.03427*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, and 1 others. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *arXiv preprint arXiv:2306.06624*.

Jimin Sun, So Yeon Min, Yingshan Chang, and Yonatan Bisk. 2024. Tools fail: Detecting silent errors in faulty tools. *arXiv preprint arXiv:2406.19228*.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand. Association for Computational Linguistics.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models: A critical investigation. *Advances in Neural Information Processing Systems*, 36.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023a. Pmc-llama: Towards building open-source language models for medicine. *arXiv preprint arXiv:2304.14454*.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023b. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023c. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*.

Hongshen Xu, Zichen Zhu, Lei Pan, Zihan Wang, Su Zhu, Da Ma, Ruisheng Cao, Lu Chen, and Kai Yu. 2024. Reducing tool hallucination via reliability alignment. *arXiv preprint arXiv:2412.04141*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Yi Yang, Yixuan Tang, and Kar Yan Tam. 2023a. Investlm: A large language model for investment using financial domain instruction tuning. *arXiv preprint arXiv:2309.13064*.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023b. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, and 1 others. 2023. mplug-docowl: Modularized multimodal large language model for document understanding. *arXiv preprint arXiv:2307.02499*.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W. Suchow, and Khaldoun Khashanah. 2023. Finmem: A performance-enhanced LLM trading agent with layered memory and character design. *arXiv preprint arXiv:2311.13743*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. 2024. ProAgent: Building proactive cooperative agents with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2023. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and 1 others. 2024a. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. 2024b. SOTOPIA: Interactive evaluation for social intelligence in language agents. *arXiv preprint arXiv:2310.11667*.

A DPO and GRPO Formulations

Direct Preference Optimization (DPO) (Rafailov et al., 2023; Ge et al., 2026) optimizes over preference pairs (z^+, z^-) of winning and losing reasoning or tool-use trajectories. Given input x, the objective increases the log-likelihood ratio of the preferred trajectory z^+ relative to z^-, regularized against a reference policy P_{θ_0}:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}\Bigg[\log \sigma\Big(\beta \log \frac{P_\theta(z^+|x)}{P_{\theta_0}(z^+|x)} - \beta \log \frac{P_\theta(z^-|x)}{P_{\theta_0}(z^-|x)}\Big)\Bigg] \tag{6}$$

where β controls the deviation from the reference policy.

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) extends DPO by sampling a group of G trajectories $\{z_1, \ldots, z_G\}$ for each input x and computing advantages relative to the group mean reward, enabling simultaneous comparison across multiple candidate reasoning or tool-use paths.