HierarchiCraft: Neural-Symbolic System to Support Agent Interactive Planning and Learning

Jieyu Zhou
Georgia Institute of Technology
Atlanta, Georgia, USA
jzhou625@gatech.edu

Jisu Kim
Georgia Institute of Technology
Atlanta, Georgia, USA
jisu.kim@gatech.edu

Baixiao Chen
Emory University
Atlanta, Georgia, USA
baixiao.chen@emory.edu

Daniel Weitekamp
Georgia Institute of Technology
Atlanta, Georgia, USA
dweitekamp3@gatech.edu

Christopher J. MacLellan
Georgia Institute of Technology
Atlanta, Georgia, USA
cmaclell@gatech.edu

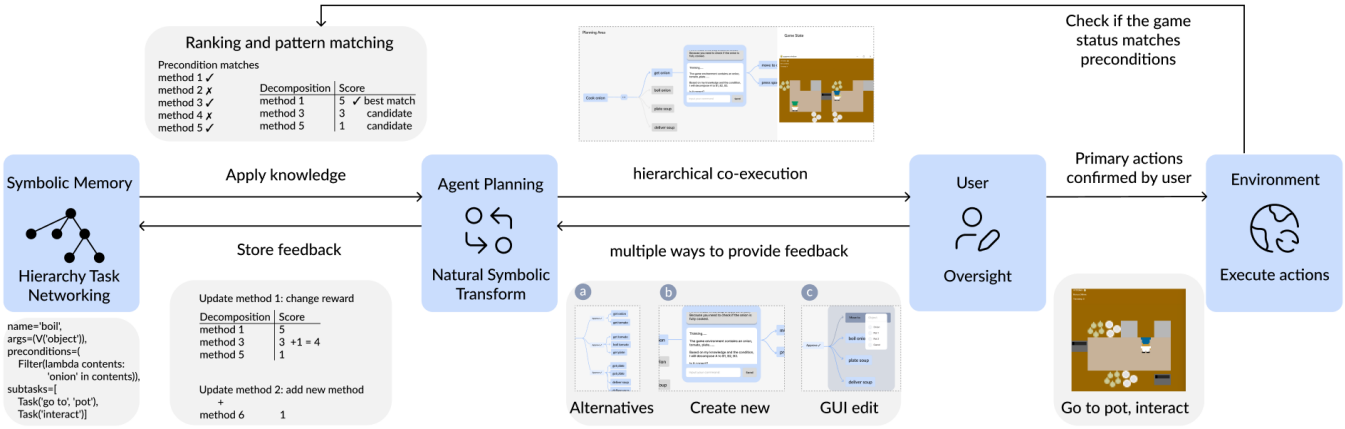

Figure 1: Overview of our neural-symbolic interactive planning framework. The agent's symbolic procedural memory and planning module form the system design layer, while the interactive interface enables users to understand, co-plan, and correct at multiple levels of abstraction.

Abstract

Large language model (LLM)–based agents are increasingly used in everyday applications, yet their reasoning remains opaque and their feedback loops brittle. Users struggle to understand how these agents act or to ensure that feedback leads to lasting improvement. We present HierarchiCraft, a neuro-symbolic system that unifies a Hierarchical Task Network (HTN) planner with an interactive interface for interpretable, controllable, and learnable agent behavior. User feedback is compiled into symbolic methods and preconditions, enabling lightweight procedural learning without fine-tuning. The interface mirrors the planner's hierarchy, supporting pauseable, stepwise execution and three feedback pathways: selecting alternative decompositions, directly editing nodes in the GUI, or authoring new methods in natural language. A formative study and comparative evaluation show that HierarchiCraft improves user understanding, oversight, and confidence, offering a design probe for error handling and continual learning in agentic AI.

ACM Reference Format:
Jieyu Zhou, Jisu Kim, Baixiao Chen, Daniel Weitekamp, and Christopher J. MacLellan. 2025. HierarchiCraft: Neural-Symbolic System to Support Agent Interactive Planning and Learning. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large language model (LLM)–based agents rely solely on model calls to generate plans and directly execute actions in the environment. These agents have rapidly entered everyday applications such as multimodal assistants [9, 44], web automation [18, 29, 30], and games [27, 54]. However, these agents remain highly error-prone, achieving only 30% accuracy on multi-step benchmarks [36, 59, 60] and therefore still require human oversight and feedback. However, achieving effective user oversight and feedback, we need to face two systemitic challenges:

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

First, users struggle to understand how LLM-based agents reason and act. LLM's next-token prediction over probability is fundamentally a black box, offering no interpretable link between reasoning and output. Even with explanations such as chain-of-thoughts [56] or explainable AI methods [25, 32], these remain surface-level observations of agent behavior. LLMs lack a true grasp of causality and logic, instead mimicking linguistic patterns from training data [40, 48, 52], which leads to hallucinations and ungrounded reasoning. The opacity pure LLM-based architecture makes it it difficult for users to identify where errors occur and provide targeted feedback.

Second, user feedback is not effectively absorbed into the agent's knowledge or reliably applied to future tasks. LLM-based agents generally learn through two mechanisms: fine-tuning [10, 42] or in-context learning [15, 46, 62]. Fine-tuning cannot cannot easily integrate small feedback loops (e.g., user corrections over one action) and risk catastrophic forgetting [43, 64]. In-context learning inherits LLM's limited reasoning and planning ability, yielding ungrounded or inconsistent reasoning steps [24], poor cross-task transferability [31], and unstable memory under long prompts or retrieval augmentation [4].

Recent HCI research has explored how users can provide feedback to AI agents, such as offering stepwise guidance on complex tasks [23, 65], visualizing the execution process [11, 13], and co-constructing the agent's workflow [2, 26, 49]. However, these systems are typically built on LLM-wrapper pipelines that call LLMs directly for planning and execution, leaving the two aforementioned challenges unresolved. Their reasoning traces remain based on probabilistic next-token prediction and thus inherently uninterpretable. Meanwhile, human feedback remains confined to correcting isolated errors in specific environment states and cannot be internalized or generalized to improve the agent's reasoning and planning over time. Consequently, user control over agent behavior remains constrained and agent still remain error-prone as feedback cannot truly improve agent performance. Therefore, to resolve these two challenges, we should not only design intuitive user interface, but also a fundamental reconsideration of how the agent itself is architected.

We present a neural-symbolic agent HierichiCraft that re-architects both the system and the interface. Unlike prior work that wraps LLMs with surface-level workflow UIs, our approach rethinks the underlying agent framework by introducing symbolic components: Hierarchical Task Network (HTN). User feedback is compiled into symbolic methods and preconditions, supporting lightweight, modular learning without fine-tuning and avoiding forgetting. In addition, HTNs save task execution methods with their preconditions as a symbolic planner, which can be reliably used as interpretation and robustly functions without hallucination. At the interface level, our design addresses three key goals. (1) Understanding: By visualizing plans as structured hierarchical decompositions, the system enables users to quickly grasp what the agent is doing and why, offering clearer task context than existing sequential, text-based displays. (2) Execution: The interface supports hierarchical, stepwise execution with pausing, allowing users to inspect or intervene at any level of abstraction rather than waiting for end-to-end autonomous runs. This layered execution design enhances transparency and controllability. (3) Correction: The system provides multiple pathways for error correction—users may select alternative decompositions, directly edit nodes in the GUI, or author new methods in natural language. These lightweight mechanisms make correction more flexible and ensure that user input is efficiently integrated into the agent's procedural knowledge.

Our contributions are:

(1) A qualitative formative study identifying user needs for hierarchical co-execution and durable feedback absorption, highlighting limits of sequential, end-to-end LLM agents.
(2) A neuro-symbolic framework that compiles user feedback into reusable procedures with explicit preconditions, avoiding fine-tuning costs and catastrophic forgetting.
(3) A tightly coupled system and interface: symbolic planning enables hierarchical visualization, pausing, and three complementary correction pathways that map directly to symbolic updates.
(4) An empirical evaluation demonstrating user preference and improved controllability over neural-only baselines, and a discussion of implications for error handling in agentic AI.

2 Related Work

We review prior work on agent learning and human oversight, outlining technical and interface-level gaps that motivate HierarchiCraft's symbolic, hierarchical design.

2.1 Agent Learning

Pure LLM-based agents generally learn through two mechanisms: parametric updates or in-context learning. However, both approaches suffer from limited robustness and interpretability, motivating the integration of symbolic components into LLM frameworks.

Parametric updates—typically implemented by fine-tuning a large base model—modify model weights through additional training to acquire new knowledge. A representative fine-tuning approach leverages reward models to guide optimization, with Human Feedback Reinforcement Learning (RLHF) as a prominent example [10, 42]. While this can theoretically enhance reasoning ability, it is brittle, computationally expensive and data-hungry. Fine-tuning on limited data often causes overfitting and instability, where new updates disrupt previously learned behaviors or even lead to catastrophic forgetting across domains [43, 64]. Moreover, the gain in correctness is not proportional to the cost in time and resources [53]. Parametric updates also lack online adaptability—they cannot incrementally incorporate small feedback loops (e.g., user corrections or contextual changes).

Consequently, recent research has shifted toward in-context learning, which adapts reasoning and planning behavior without modifying model weights. Frameworks such as ReAct [62], Reflexion [46], and RAP [15] extend LLM reasoning by interleaving reasoning traces and actions, maintaining episodic memory for self-reflection, or constructing reasoning trees to balance exploration and exploitation. Although in-context learning supports incremental adaptation and avoids costly retraining, the robustness problem still remains unsolved. LLMs appear capable of reasoning through contextual cues, yet this ability primarily reflects linguistic fluency rather than genuine algorithmic reasoning [40, 48]. As a result, ReAct often produces ungrounded or inconsistent reasoning steps that misalign with the environment state [24], and Reflexion improves

performance only under specific conditions—when initial responses are unreliable and task difficulty is high—showing limited generality [21, 31]. Moreover, their reliance on extended context often necessitates retrieval-augmented generation (RAG) [28], which introduces additional failure modes, including missing or misranked documents, poor consolidation, and incorrect specificity [4].

Beyond robustness, both parametric and in-context approaches face a fundamental limitation from the human perspective. Their probabilistic formulation differs fundamentally from how humans reason and plan [7, 50]. Even with the aid of explainability tools, this black-box nature cannot be fully resolved—the underlying process remains opaque and non-causal, making it difficult for humans to fully comprehend or predict model behavior. Moreover, their probabilistic generation often results in hallucinations and logically inconsistent outputs [17, 19, 52], further undermining human trust in their reasoning.

To address these challenges, recent efforts have incorporated symbolic components into LLM frameworks—using symbolic toolkits for logical computation [22], coupling LLMs with symbolic reasoning engines [61], or employing symbolic checkers to validate candidate actions [40]. However, these systems remain fully autonomous, optimizing via trial and error without human-in-the-loop feedback. We next review HCI systems for human oversight, showing that many adopt visualization-first designs over action-centric LLM wrappers, which rarely realize durable learning.

2.2 Human Oversight in Agents

The rapid proliferation of AI agents has reignited a long-standing HCI debate over automation and human control [16, 33, 37, 47]. Most commercial agents nowadays executing tasks end-to-end with little to no user oversight [1, 38, 41], making it hard for user to pause the agent and correct the wrong step. HCI researchers have proposed a wide range of tools for human oversight in agent planning and execution across various domains, such as creativity [14, 49], data analysis [13, 23] coding[45]. Across these studies, there are two visualization paradigms: chat-first and GUI-first. The chat-first systems enables user to modify agent behavior through conversation interaction [11, 13, 23, 26]. While some systems integrate supplementary visual widgets for user to better understand the process, the chat remains the central control medium. In contrast, the GUI-first paradigm represents each action as a node on the canvas, allowing users to directly drag connections, delete or add nodes to revise the plan [2, 12, 49, 63, 65]. Among these GUI-first designs, hierarchical visualization is commonly adopted approach. For instance, WaitGPT visualizes hierarchical data transformation trajectories [58], AmbigChat structures conversations hierarchically to support efficient search and disambiguation [35], and Graphlogue presents layered visualizations of conceptual relationships [20].

Despite these advances, many systems in HCI field lack learning ability and are brittle in complex, changing environments. This limitation arises because most systems are visualization-oriented front ends built on an LLM-wrapper pipeline (i.e., decomposing tasks and prompting an LLM to handle subtasks) without mechanisms for continual or cross-episode learning. Two factors contribute to this problem. First, few system implement the conventional learning approaches disucssed in Section 2.1 (parametric updates and in-context learning). Many systems' prompting is shallow and ad hoc without in-context schemas or structured error traces, so behavior does not improve across episodes. Second, an agent architecture generally comprises profile, memory, planning, and action modules [50, 55, 66]. These systems emphasize only the action component, omitting explicit memory and planning. As a result, knowledge is neither retained nor consolidated, and interface edits cannot be validated against an executable internal model. A partial exception is VAL, which compiles natural-language instructions into symbolic structures [27]; however, its interaction is primarily chat-driven and does not foreground visualization of the executable plan state.

HierarchiCraft bridges frontend and backend by making a symbolic, HTN-based representation the substrate of the system. The same hierarchical structure that governs planning and execution is rendered to the UI, so user edits (e.g., refine, reorder, replace, prune) map directly to well-typed plan operations rather than ad-hoc prompt patching. Rather than starting from an interface wish list and then "stitching" prompts to approximate it, we co-design the agent's internal structure and the interface, achieving a tight alignment between architecture and visualization while enabling durable learning mechanisms.

3 Formative Study

We conducted a formative study with 10 participants to understand how users perceive and interact with current agentic interfaces. We selected five tasks to represent a broad spectrum of domains, including everyday activities (travel planning and online shopping), office workflows (document process and image editing), and virtual environments (game playing), drawing from established AI agent benchmarks [51, 57, 59, 60, 68]. We selected three agents that have different visualization rationales: (1) actions as clickable nodes with environment screenshots, (2) actions embedded in replay, and (3) replay only. We randomly assigned tasks to these agents. Each session lasted approximately 45 minutes experiencing these five tasks. Participants were asked to think aloud, verbalizing their judgments about correctness and describing how they identified errors. After task completion, we conducted semi-structured interviews to explore participants' expectations and desired experiences with agent interfaces. All sessions were recorded, transcribed, and thematically analyzed [6, 8]. Two researchers collaboratively coded the transcripts and refined emerging themes about how to help users better understand agent reasoning and facilitate learning from human feedback.

3.1 Interpretable Planning and Action

The hallucination of LLM-based agent makes participants difficult to understand its behavior and fail to provide feedback. LLM-based agents generates redundant and verbose reasoning traces but lack of rationale. One participant remarked, "Since it has a lot of random code execution involved, I have no idea what to look for." (P5) This lack of interpretability was especially problematic when agents made mistakes. The most frequent error we observed was a mismatch between the agent's described activity and its actual execution. For example, in a shopping-cart task, the agent's log claimed that all four selected items were added, but only two appeared in the

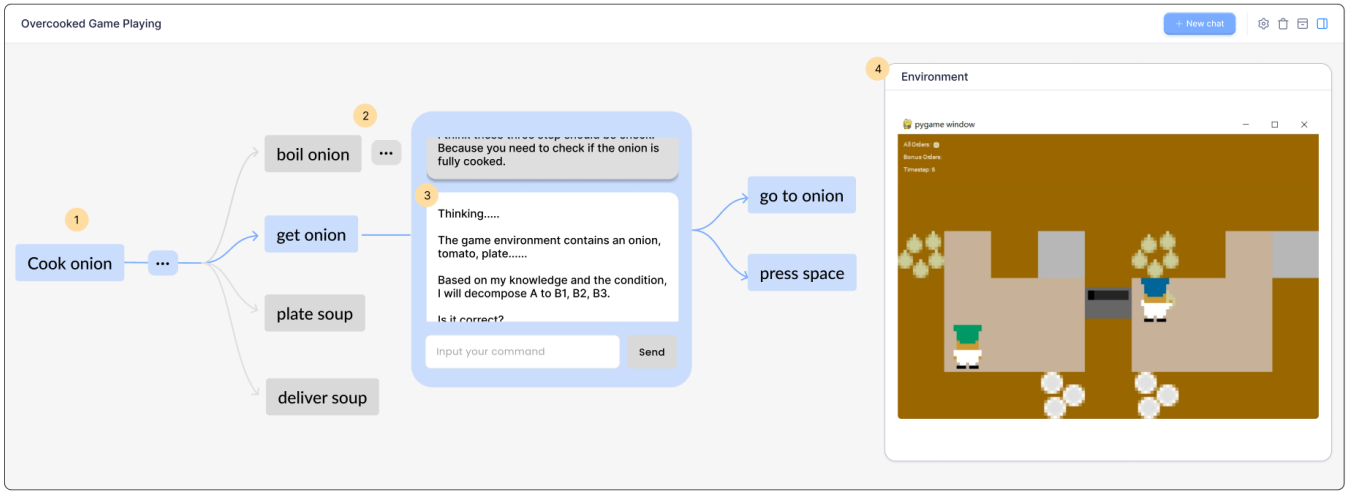

Figure 2: Hierarchical visualization and co-execution in HierarchiCraft. (1) The task hierarchy allows users to pause execution at any level. (2) Executed subtasks are hidden while the current node is highlighted to focus attention. (3) The agent provides interpretable reasoning grounded in symbolic preconditions. (4) The environment is jointly perceived by both the agent and the user, enabling verification and shared situational awareness.

cart. Such hallucinated perceptions of successful completion reflect the ungrounded and inconsistent reasoning frequently observed in LLM-based agents [40, 48, 52]. As one participant noted, "It hid the mistake from me by saying conflicting information. I think I would believe what it said and then be disappointed when my groceries arrived." (P4) Users found these hallucinations difficult to detect and even harder to correct, highlighting the need for agents to explicitly verify task completion through symbolic checking mechanisms.

Current agents does not have planning phase, executing user command immediately. Users cannot predict what the agent will do next and do not know the agent's intention. Participants expressed a strong preference for seeing an overview of the plan before execution to check their commands were understood by agents. Early plan alignment thus helps users identify potential issues and avoid wasted effort [3, 67].

Design Goal: Generate grounded, interpretable plans that accurately reflect the real-world environment and mitigate hallucination.

3.2 Reliable Control

adoption strategy Agents frequently failed to improve after user corrections. For instance, in the hotel booking task, the target dates were July 4–12, but the agent initially selected July 9–12. After being corrected, it instead chose July 4–August 12. Even when the agent eventually produced the correct dates, users still doubted whether it had truly learned from feedback. But users still feel confused and did not know if the agent can perform well when it encounters the same situation next time. Participant P8 commented, "*The agents should verify understanding [after I gave feedback]. Its clarifying questions can make it easy to see when they are confusing.*" Agents should respond to user correction by explicitly summarizing changes in the world state or clarifying residual ambiguities, to signal their ability to learn from users. Furthermore, when encountering similar situations in the future, agents should leverage memory to retrieve prior user feedback, adapt their decision-making accordingly, and make visible which parts of their reasoning draw on previous interactions.

Design Goal: Enable agents to absorb user feedback into their knowledge and communicate signals of learning and understanding.

3.3 Flexible Error Detection and Correction

Current agents typically visualize plans as lengthy sequential text lists, causing users to become disoriented and making it difficult to locate specific errors. One participant noted, "If I agree with the way you've laid out the tasks, I'd want to drill down specifically into the subtasks where I think the error is." (P4) To support efficient error detection, agents should visualize plans at multiple levels of abstraction, allowing users to progressively hide or reveal details as needed. This design aligns with the principle of progressive disclosure, which advocates for presenting information incrementally to reduce cognitive load [5, 39].

Participants emphasized the importance of pausing execution to provide timely feedback. Current end-to-end execution pipelines make it impossible to intervene once an early error occurs, often leading to error propagation or repetitive loops. Allowing users to interrupt or correct the process mid-execution can prevent unnecessary actions and reduce wasted API calls [34, 67]. Furthermore, agents should support multiple correction modalities. "If the task is simple, I will directly correct it in the action node. If the task is difficult and the error is complex, I will explain in detail to the agent how to correct it." (P10) The type and complexity of the task often influence users' feedback strategies.

Design Goal: Visualize plans at multiple levels of abstraction and allow users to pause and modify execution through flexible feedback mechanisms

4 System Design

HierarchiCraft couples a neuro-symbolic agent with an interaction model that keeps planning interpretable, execution controllable, and learning lightweight. The architecture centers on a Hierarchical Task Network (HTN) planner and a symbolic procedural memory; the interface renders this same hierarchy so that user oversight and edits are always type-safe plan operations rather than prompt patching. Figure 1 summarizes the loop: an LLM produces an initial outline, the system compiles it into symbolic methods with preconditions, the user supervises hierarchical co-execution, and feedback is stored back into symbolic memory to affect subsequent planning without fine-tuning. This section details the interaction flow, representation, visualization, feedback mechanisms, and the planner's ranking and learning procedure that together realize the design goals derived from our formative study.

4.1 Interaction Flow

The system maintains a closed loop across user, planner, and environment. Given a user goal, the agent first enters a *planning phase* that interprets the LLM outline and proposes a decomposition; before any environment actuation, the plan is surfaced to the interface for alignment. During *co-execution*, users approve subtasks at chosen granularity; confirmed nodes trigger primitive actions while unconfirmed branches remain pending. Users may pause, inspect, or refocus on a sub-decomposition, and revise the structure at any moment. These operations are validated against preconditions and the current world state. In the *feedback phase*, corrections—selection among candidates, natural language authoring of new methods, or GUI edits—are compiled back into symbolic memory and update the planner's ranking signals so that later episodes retrieve and prefer the newly successful method under matching conditions. This loop delivers transparent intent, interruptibility, and durable absorption of user input, capabilities difficult to achieve in sequential end-to-end agents.

4.2 Symbolic Representation

Procedural knowledge is encoded as HTN *methods* with typed arguments, preconditions over symbolic world predicates, and ordered subtasks that expand recursively to primitives. A symbolic parser maps natural-language plans and predefined domain knowledge into this representation, emitting well-typed operators and filters that can be checked against the live game state before execution. Because preconditions are explicit and executable, the planner can reject inconsistent proposals early and surface only grounded alternatives to the user, providing a stable substrate for interpretation and control while avoiding hallucinated steps.

4.3 Hierarchical Visualization

The interface renders the planner's internal HTN one-to-one: each node corresponds to a subgoal, its children to a candidate decomposition, and leaves to primitive actions. Users progressively disclose or hide detail, shift focus to the active node, and pause or resume execution at any level; confirmations flow through the hierarchy into environment actions, while the current state is mirrored on the canvas. This direct mirroring eliminates the gulf between reasoning traces and actual execution, reduces disorientation from long

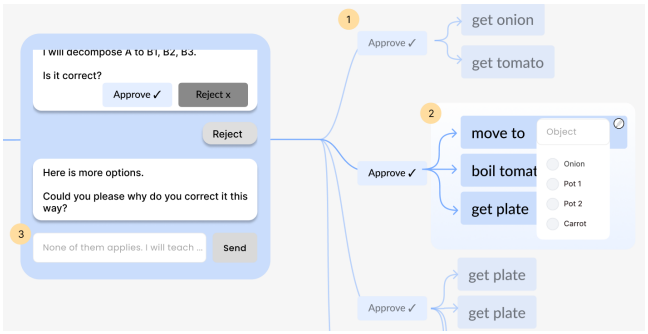

Figure 3: Three feedback modes in HierarchiCraft: (1) selecting more options, (2) directly editing nodes in the GUI, and (3) chat-based authoring of new methods, all compiled into symbolic memory for future planning.

textual logs, and enables precise localization of errors within the plan tree. The visualization implements the study-elicited needs for overview-first planning, mid-course intervention, and multilevel inspection.

4.4 Flexible Feedback Mechanism

HierarchiCraft unifies three feedback pathways, each compiled into the same symbolic substrate. (1) *Alternative selection:* when multiple decompositions satisfy preconditions, the planner ranks them and surfaces candidates; user choice both drives execution and updates the corresponding method's score. (2) *Natural-language authoring:* users describe missing strategies, which are parsed into new methods and inserted into memory with initial scores. (3) *Direct GUI edits:* users refine, reorder, or replace nodes, and edits are translated into validated method updates. All three modes affect the current episode immediately and are stored to support retrieval under similar contexts later, addressing the observed gap where corrections failed to generalize across tasks.

4.5 Interpretable Planning and Learning

Planning proceeds by *pattern matching and ranking*. Given a target task and state, the planner filters methods whose preconditions match, computes a score from symbolic context features and learned rewards, and proposes the top-ranked decomposition. During co-execution, outcome signals (success, failure, or explicit user approvals) adjust method scores, while users may attach new preconditions that gate applicability. Because updates are local, modular, and symbolic—not gradient-based—learning is lightweight, avoids catastrophic forgetting, and yields explanations directly from the method and its preconditions. Over time, the memory accrues domain-appropriate alternatives and discriminative guards, improving robustness without sacrificing transparency.

References

[1] Anthropic. 2024. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. https://www.anthropic.com/news/3-5-models-and-computer-use. Accessed: July 2025,.

[2] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L Glassman. 2024,. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the 2024 CHI Conference on*

Human Factors in Computing Systems. 1–18.

[3] Gagan Bansal, Jennifer Wortman Vaughan, Saleema Amershi, Eric Horvitz, Adam Fourney, Hussein Mozannar, Victor Dibia, and Daniel S Weld. 2024,. Challenges in human-agent communication. *arXiv preprint arXiv:2412.10380* (2024,).

[4] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024,. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 194–199.

[5] Benjamin B Bederson and James D Hollan. 1994,. Pad++ a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*. 17–26.

[6] Hugh Beyer and Karen Holtzblatt. 1999. Contextual design. *interactions* 6, 1 (1999), 32–42.

[7] Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and TP Singh. 2024. Neuro-symbolic artificial intelligence: a survey. *Neural Computing and Applications* 36, 21 (2024), 12809–12844.

[8] Virginia Braun and Victoria Clarke. 2012. *Thematic analysis*. 57–71.

[9] Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, et al. 2024,. PARTNR: A Benchmark for Planning and Reasoning in Embodied Multi-agent Tasks. *arXiv preprint arXiv:2411.00081* (2024,).

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017,. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017,).

[11] Adam J Coscia, Shunan Guo, Eunyee Koh, and Alex Endert. 2025,. OnGoal: Tracking and Visualizing Conversational Goals in Multi-Turn Dialogue with Large Language Models. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–18.

[12] Anindya Das Antar, Somayeh Molaei, Yan-Ying Chen, Matthew L Lee, and Nikola Banovic. 2024,. VIME: Visual Interactive Model Explorer for Identifying Capabilities and Limitations of Machine Learning Models for Sequential Decision-Making. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–21.

[13] Will Epperson, Gagan Bansal, Victor C Dibia, Adam Fourney, Jack Gerrits, Erkang Zhu, and Saleema Amershi. 2025,. Interactive debugging and steering of multi-agent ai systems. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. 1–15.

[14] KJ Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S Weld, Amy X Zhang, and Joseph Chee Chang. 2024,. Cocoa: Co-planning and co-execution with ai agents. *arXiv preprint arXiv:2412.10999* (2024,).

[15] Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. [n. d.]. Reasoning with Language Model is Planning with World Model. In *NeurIPS 2023 Workshop on Generalization in Planning,*.

[16] Eric Horvitz. 1999,. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 159–166.

[17] Jie Huang and Kevin Chen-Chuan Chang. 2022,. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403* (2022,).

[18] Faria Huq, Jeffrey P Bigham, and Nikolas Martelaro. 2023,. " What's important here?": Opportunities and Challenges of Using LLMs in Retrieving Information from Web Interfaces. *arXiv preprint arXiv:2312.06147* (2023,).

[19] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[20] Peiling Jiang, Jude Rayan, Steven P Dow, and Haijun Xia. 2023,. Graphologue: Exploring large language model responses with interactive diagrams. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*. 1–20.

[21] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022,. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221* (2022,).

[22] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. 2022,. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445* (2022,).

[23] Majeed Kazemitabaar, Jack Williams, Ian Drosos, Tovi Grossman, Austin Zachary Henley, Carina Negreanu, and Advait Sarkar. 2024,. Improving steering and verification in AI-assisted data analysis with interactive task decomposition. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–19.

[24] Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. 2025,. ReflAct: World-Grounded Decision Making in LLM Agents via Goal-State Reflection. *arXiv preprint arXiv:2505.15182* (2025,).

[25] Sunnie SY Kim, Elizabeth Anne Watkins, Olga Russakovsky, Ruth Fong, and Andrés Monroy-Hernández. 2023,. " help me help the ai": Understanding how explainability can support human-ai interaction. In *proceedings of the 2023 CHI conference on human factors in computing systems*. 1–17.

[26] Lane Lawley and Christopher Maclellan. 2024,. Val: Interactive task learning with gpt dialog parsing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–18.

[27] Lane Lawley and Christopher J. MacLellan. 2024,. VAL: Interactive Task Learning with GPT Dialog Parsing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. doi:10.1145/3613904.3641915

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020,. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020,), 9459–9474.

[29] Amanda Li, Jason Wu, and Jeffrey P Bigham. 2023,. Using LLMs to Customize the UI of Webpages. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–3.

[30] Toby Jia-Jun Li, Amos Azaria, and Brad A Myers. 2017,. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. 6038–6049.

[31] Yanhong Li, Chenghao Yang, and Allyson Ettinger. 2024,. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. *arXiv preprint arXiv:2404.09129* (2024,).

[32] Q Vera Liao and Kush R Varshney. 2021,. Human-centered explainable ai (xai): From algorithms to user experiences. *arXiv preprint arXiv:2110.10790* (2021,).

[33] Henry Lieberman. 1997,. Autonomous interface agents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 67–74.

[34] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2023,. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of machine learning research* 24, 253 (2023,), 1–15.

[35] Jiaju Ma, Lei Shi, Kenneth Aleksander Robertsen, and Peggy Chi. 2025,. AmbigChat: Interactive Hierarchical Clarification for Ambiguous Open-Domain Question Answering. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–18.

[36] Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024,. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. *Advances in Neural Information Processing Systems* 37 (2024,), 94871–94908.

[37] Pattie Maes. 1995. Agents that reduce work and information overload. In *Readings in human–computer interaction*. Elsevier,, 811–821.

[38] Manus. 2024. Leave it to Manus: General AI Agent for Work and Life. https://manus.im/home. Accessed: July 2025,.

[39] Donald A. Norman and Stephen W. Draper. 1986. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., USA,.

[40] Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. 2021,. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems* 34 (2021,), 25192–25204.

[41] OpenAI. 2025. Introducing ChatGPT agent: bridging research and action. https://openai.com/index/introducing-chatgpt-agent. Accessed: July 2025,.

[42] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022,. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022,), 27730–27744.

[43] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71. doi:10.1016/j.neunet.2019.01.012

[44] rabbit research team. 2023,. Learning human actions on computer applications. https://rabbit.tech/research

[45] Yijia Shao, Vinay Samuel, Yucheng Jiang, John Yang, and Diyi Yang. 2024,. Collaborative gym: A framework for enabling and evaluating human-agent collaboration. *arXiv preprint arXiv:2412.15701* (2024,).

[46] 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 (2023,), 8634–8652.

[47] Ben Shneiderman and Pattie Maes. 1997. Direct manipulation vs. interface agents. *interactions* 4, 6 (1997), 42–61.

[48] Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024,. Chain of thoughtlessness? an analysis of cot in planning. *Advances in Neural Information Processing Systems* 37 (2024,), 29106–29141.

[49] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. 2024,. Luminate: Structured generation and exploration of design space with large language models for human-ai co-creation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–26.

[50] Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2023,. Cognitive architectures for language agents. *Transactions on Machine Learning Research* (2023,).

[51] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023,. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances*

in Neural Information Processing Systems 36 (2023,), 38975–38987.

[52] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2024,. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems* 36 (2024,).

[53] Karthik Valmeekam, Kaya Stechly, Atharva Gundawar, and Subbarao Kambhampati. 2025,. A systematic evaluation of the planning and scheduling abilities of the reasoning model o1. *Transactions on Machine Learning Research* (2025,).

[54] 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:2305.16291 [cs.AI,]

[55] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[56] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022,. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022,), 24824–24837.

[57] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024,. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622* (2024,).

[58] Liwenhan Xie, Chengbo Zheng, Haijun Xia, Huamin Qu, and Chen Zhu-Tian. 2024,. Waitgpt: Monitoring and steering conversational llm agent in data analysis with on-the-fly code visualization. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[59] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024,. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems* 37 (2024,), 52040–52094.

[60] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. 2024. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. arXiv:2412.14161 [cs.CL] https://arxiv.org/abs/2412.14161

[61] Zhun Yang, Adam Ishay, and Joohyung Lee. 2023,. Coupling Large Language Models with Logic Programming for Robust and General Reasoning from Text. In *Findings of the Association for Computational Linguistics: ACL 2023*. 5186–5219.

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

[63] Ryan Yen and Jian Zhao. 2024,. Memolet: Reifying the reuse of user-ai conversational memories. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–22.

[64] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems* 43, 6 (2025), 1–47.

[65] Yuheng Zhao, Junjie Wang, Linbin Xiang, Xiaowen Zhang, Zifei Guo, Cagatay Turkay, Yu Zhang, and Siming Chen. 2024. Lightva: Lightweight visual analytics with llm agent-based task planning and execution. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[66] Jieyu Zhou and Christopher MacLellan. 2024,. Improving Interface Design in Interactive Task Learning for Hierarchical Tasks based on a Qualitative Study. In *Adjunct Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–3.

[67] Jieyu Zhou, Aryan Roy, Sneh Gupta, Daniel Weitekamp, and Christopher J MacLellan. 2025,. When Should Users Check? A Decision-Theoretic Model of Confirmation Frequency in Multi-Step AI Agent Tasks. *arXiv preprint arXiv:2510.05307* (2025,).

[68] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023,. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023,).