AGENT MODELS: INTERNALIZING CHAIN-OF-ACTION GENERATION INTO REASONING MODELS

Yuxiang Zhang[†], Yuqi Yang[†], Jiangming Shu, Xinyan Wen & Jitao Sang[*]
School of Computer Science and Technology
Beijing Jiaotong University
Beijing, China
{yuxiangzhang, yqyang, jiangmingshu, xinyanwen, jtsang}@bjtu.edu.cn

ABSTRACT

Traditional agentic workflows rely on external prompts to manage interactions with tools and the environment, which limits the autonomy of reasoning models. We position *Large Agent Models (LAMs)* that internalize the generation of *Chain-of-Action (CoA)*, enabling the model to autonomously decide when and how to use external tools. Our proposed AutoCoA framework combines supervised fine-tuning (SFT) and reinforcement learning (RL), allowing the model to seamlessly switch between reasoning and action while efficiently managing environment interactions. Main components include step-level action triggering, trajectory-level CoA optimization, and an internal world model to reduce real-environment interaction costs. Evaluations on open-domain QA tasks demonstrate that AutoCoA-trained agent models significantly outperform ReAct-based workflows in task completion, especially in tasks that require long-term reasoning and multi-step actions. Code and dataset are available at `https://github.com/ADaM-BJTU/AutoCoA`.

1 INTRODUCTION

OpenAI has outlined five progressive stages on the path to Artificial General Intelligence (AGI). The first stage, characterized as **Chatbot**, is exemplified by Large Language Models (LLMs) like *GPT-3.5* and *GPT-4* OpenAI (2023). The second stage, termed **Reasoner**, introduces Large Reasoning Models (LRMs) such as *o1* OpenAI (2024) and *o3*. Recently, OpenAI released *Operator* OpenAI (2025a) and *Deep Research* OpenAI (2025b), signaling the arrival of the third stage: **Agent**. These systems reportedly combine reasoning with autonomous tool usage, enabling independent execution of multi-round workflows by interacting with the real-world environment.

It is believed that the technology behind *Operator* and *Deep Research* is not merely integrating existing LLMs or LRMs with agentic workflows (e.g., ReAct Yao et al. (2022), Reflexion Shinn et al. (2023)). Instead, it represents a further upgrade in model capabilities: the new models are capable of long-term planning, tool manipulation, and environmental interaction. Taking *Deep Research* as an example, most currently available open-source versions, such as *Open Deep Research* Team (2025) by Hugging Face, primarily rely on agentic workflows to orchestrate task planning and tool usage. The switching between thought and action is triggered by preset workflows, making it a "passive" behavior. In contrast, OpenAI's Deep Research enhances the multi-turn tool usage capability of reasoning models. The switching between thought and action is based on the model's inherent behaviors, where the model "actively" decides when and how to take action [1]. This allows for a

---

[*]Corresponding author.

[1]† Equal contribution.

[1] In an agentic workflow employing reasoning models, the agent also triggers actions based on its own behavior during reasoning, but this switching is only based on the preceding context. In contrast, the model behind OpenAI's Deep Research has undergone end-to-end fine-tuning on the basis of reasoning models for tool usage capabilities. It considers not only the preceding context but also the subsequent context, explicitly trained with task completion as the optimization objective.

CoT prompting

prompting-based

+ task planning
(Chain-of-Thought, CoT)

learning-based

LLM
GPT-4,
DeepSeek-V3

LRM
o1,
DeepSeek-R1

Agentic workflow

prompting-based

+ tool usage
(Chain-of-Action, CoA)

learning-based

LAM
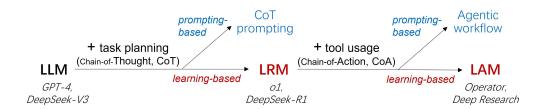*Operator,*
Deep Research

Figure 1: The path from Large Language Model (LLM) to Large Reasoning Model (LRM) and the forthcoming Large Agent Model (LAM).

longer chain of thought-action loops, enabling the completion of more complex knowledge search tasks. We believe this type of **tool-augmented reasoning model** deserves a distinct name, and we propose calling it Large Agent Models (LAMs).

Task planning and tool usage are two core capabilities of an Agent. The reasoning model internalizes the capability of task planning, i.e., generating Chain-of-Thought (CoT). By further enhancing and internalizing the capability of tool usage, i.e., generating *Chain-of-Action (CoA)*, the reasoning model evolves into an **Agent model**.

It is helpful to review the development of reasoning models to understand the relationship between agentic workflow and agent model. As illustrated in Fig. 1, a reasoner needs to add the ability to generate Chain-of-Thought (CoT) on top of the System-1 LLM. One approach is through prompting, leveraging in-context learning to "force" the model to output multi-step reasoning, commonly referred as *CoT prompting* Wei et al. (2022). Another approach is learning-based Zhang et al. (2024a); Zeng et al. (2024), whether through SFT alone, a combination of SFT followed by RL, or direct RL. This results in reasoning models like o1 and DeepSeek-R1. Unlike CoT prompting relying on external guidance to elicit multi-step reasoning, reasoning models learn the logics between thinking steps and internalize the CoT generation capability as a "active" model behavior.

Similarly, agents need to move beyond CoT reasoning to CoA generation, a paradigm dynamically interleaving thought and action. While *agentic workflow* relies on prompting-based methods to simulate this interplay, agent model learns the coupled CoT and CoA patterns into its behavior, resulting in stronger logical coherence and seamless transitions between thought and action. The model initiates tool usage only when necessary and makes context-aware decisions on how to execute it. Harnessing tools to embody in digital/physical environments, agent models effectively bridge the gap between reasoning and action.

In fact, this paradigm shift from prompting to learning aligns with AI's decades-long evolution from handcrafted rules to data-driven learning: from expert systems to classical machine learning and then to deep learning. Patterns learned from data exhibit better generalization and greater flexibility. We have already witnessed that reasoning models like OpenAI o1 and DeepSeek R1 have marginalized the role of explicit prompting, as they can better understand user intent and proactively make plans. Similarly, the shift from agentic workflows to agent models, which internalize CoA generation, is expected to gradually replace the static and scripted workflows.

In this report, we introduce the recent attempt to enhance the multi-turn tool usage capability of reasoning models, that is, to interleave CoA generation within the CoT trajectories. Our preliminary study focuses on small-scale reasoning models, using search action as a testbed for open-domain question answering (QA) tasks. Initial results demonstrate that internalizing CoA generation substantially improves task completion rates compared to agentic workflows. Future work will involve experimental validation on larger models, more tool types, and a wider range of open-ended tasks.

2 AGENT MODEL

2.1 DEFINITION AND FORMALIZATION

Agent models are enhanced with the ability to use tools. This requires it to extend beyond its own reasoning behaviors and interact with the external environment. Fig. 2 compares the interaction
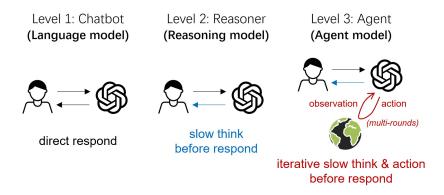
Figure 2: Interaction paradigm: Chatbot vs. Reasoner vs. Agent.

paradigms from chatbots to agent models. Both Chatbot and Reasoner focus exclusively on binary interactions between human and model. The difference lies in that reasoner-layer models engage in slow, deliberate reasoning before responding. While, the third layer, Agent, requires the model to think and act simultaneously, which takes the external environment into consideration. This creates a ternary structure of human-model-environment. Action, in this case, refers to interacting with the environment through tool usage to obtain feedback. After multiple rounds of thought, action, and observation, the final response is generated.

This applies to both Operator and Deep Research: Operator's environment consists of operating systems and applications, and Deep Research currently operates within the environment of web-based information and data resources.

We formally define *Agent Model* as follows:

Definition 1. *An **Agent Model** is a generative model built upon a reasoning model, enhanced through end-to-end task-oriented tool-augmented training. It produces sequences of interleaved reasoning (Chain-of-Thought) and action (Chain-of-Action) steps, where each action invokes a tool* [2] *to interact with the external environment. Observations from these interactions guide subsequent reasoning and actions until the task is completed.*

The inference of agent model can be formalized as a partially observable Markov decision process (POMDP), where the state is defined as $[\mathbf{s}_0, \mathbf{tc}, \mathbf{x}_{1:n}]$, comprising: (1) $\mathbf{s}_0$: the initial environment state (e.g., the state of a digital or physical system), (2) $\mathbf{tc}$: the task context (e.g., a user prompt), (3) $\mathbf{x}_{1:n}$: the sequence generated so far, combining tokens emitted by the model and responses from the environment. Actions in this POMDP are the generation of the next token $x_{n+1} \sim \pi_\theta(\cdot \mid \mathbf{s}_0, \mathbf{tc}, x_{1:n})$, where π_θ is the policy parameterized by θ.

At each step, the policy π_θ can generate one of the following types of tokens:

- ⟨think⟩: Initiates or continues a reasoning step, contributing to the Chain-of-Thought (CoT), which represents internal deliberation or planning. Following this token, a reasoning sequence $\mathbf{r} = [r_1, r_2, \ldots, r_k]$ is generated, where each r_i is a token in the reasoning process.

- ⟨action⟩: Invokes a tool to interact with the environment, contributing to the Chain-of-Action (CoA). This token signals the transition from thought to action, deciding *when to action*. Following this token, the model internally generates or acquires from environment:
  - *Action type*: specifies the tool from tool library $\tau \in \mathcal{T}$ (e.g., "search"),
  - *Action parameters*: $\mathbf{p}$, defines how to execute the tool, deciding *how to action* (e.g., detailed search query),
  - *Response by action invocation*: fetches the environment's response $\mathbf{o} = \tau(\mathbf{p})$, analogous to observations in ReAct. Note that this response is not generated by policy π_θ

---

[2] Tools can be broadly categorized into two types: those for accessing external resources, such as real-time information and specialized databases via web browsers and API interfaces, and those for professional processing, such as complex computation and data analysis softwares.

3

but returned by interacting with the environment. **o** will appended to the trajectory, providing context for subsequent turns.

- ⟨answer⟩: Outputs the final response, indicating task completion or termination due to exceeding a predefined budget. Following this token, an answer sequence $\mathbf{a} = [a_1, a_2, \ldots, a_m]$ is generated, representing the task solution.

The inference pipeline of agent model is shown in Algorithm 1, outlining the step-by-step process by which the agent model generates its trajectory. The algorithm iteratively generates tokens based on the current state, deciding whether to engage in further reasoning, invoke a tool, or conclude with an answer. This pipeline ensures that the model alternates between internal reasoning (CoT) and external actions (CoA), allowing it to dynamically adapt its strategy based on environmental feedback. The loop continues until the task is completed or a predefined limit on reasoning-action iterations/contextual length is reached.

We can see that, unlike agentic workflows that rely on external prompts or scripts to manage interactions, the agent model internalizes the logic of tool usage as part of its learned behavior, enabling autonomous and context-aware decision-making for tool invocation.

Algorithm 1 Agent Model Inference Pipeline

Require:
    Policy model π_θ, tool library $\mathcal{T}$,
    Task context $\mathbf{tc}$, max action steps T, max contextual length N
Ensure:
    Final answer $\mathbf{a}$

1: Initialize: $t \leftarrow 0$, $n \leftarrow \#(\mathbf{tc})$, $x_{1:n} \leftarrow \mathbf{tc}$,
2: **while** $t < T$ **and** $n < N$ **and** ⟨answer⟩ not in $x_{1:n}$ **do**
3:     Generate token: $x_{n+1} \sim \pi_\theta(\cdot \mid x_{1:n})$
4:     $x_{1:n+1} \leftarrow x_{1:n} \circ x_{n+1}$, $n \leftarrow n+1$
5:     **if** $x_n = $ ⟨think⟩ **then**   ▷ Generate CoT
6:         **while** next token $\notin \{$⟨action⟩, ⟨answer⟩$\}$ **do**
7:             $r \sim \pi_\theta(\cdot \mid x_{1:n})$
8:             $x_{1:n+1} \leftarrow x_{1:n} \circ r$, $n \leftarrow n+1$
9:         **end while**
10:     **else if** $x_n = $ ⟨action⟩ **then**   ▷ Generate CoA
11:         $t \leftarrow t+1$
12:         Generate $\mathbf{c} \sim \pi_\theta(\cdot \mid x_{1:n})$, $k = \#(\mathbf{c})$ ▷ Generate action type
13:         $x_{1:n+k} \leftarrow x_{1:n} \circ \tau$, $n \leftarrow n+k$
14:         Select tool: $\tau \leftarrow \mathcal{T}[\mathbf{c}]$ ▷ *map action type tokens to specific tool*
15:         Generate $\mathbf{p} \sim \pi_\theta(\cdot \mid x_{1:n})$, $k = \#(\mathbf{p})$ ▷ Generate action parameter
16:         $x_{1:n+k} \leftarrow x_{1:n} \circ \mathbf{p}$, $n \leftarrow n+k$
17:         Execute: $\mathbf{o} \leftarrow \tau(\mathbf{p})$, $k = \#(\mathbf{p})$ ▷ Invoke tool and acquire response
18:         $x_{1:n+k} \leftarrow x_{1:n} \circ \mathbf{o}$, $n \leftarrow n+k$
19:     **else if** $x_n = $ ⟨answer⟩ **then**
20:         Generate $\mathbf{a} \sim \pi_\theta(\cdot \mid x_{1:n})$, $n \leftarrow n+1$ ▷ Generate answer sequence
21:         **return** $\mathbf{a}$
22:     **end if**
23: **end while**
24: **if** ⟨answer⟩ $\notin x_{1:n}$ **then**   ▷ Force termination and answer
25:     $x_{1:n} \leftarrow x_{1:n} \circ$ ⟨answer⟩
26:     Generate $\mathbf{a} \sim \pi_\theta(\cdot \mid x_{1:n})$
27: **end if**
28: **return** $\mathbf{a}$

2.2 CHALLENGES

The development of agent model requires end-to-end training on top of reasoning models to enhance its capabilities in tool usage and interaction with the external environment. This presents two fundamental challenges.

Challenge 1: Balancing Reasoning and Action. Reasoning models excel at generating Chains-of-Thought (CoT). Fine-tuning these models to enhance action execution capabilities poses a common problem: the risk of forgetting the model's CoT-generation ability Zhang et al. (2024b). More critically, the key to interleaving CoT with Chain-of-Action (CoA) lies in teaching the model to trigger actions at appropriate times within the reasoning process. This ensures that reasoning and action complement each other effectively to jointly accomplish the task.

Challenge 2: Managing External Environment Interaction. Interacting with the external environment through tool invocations may result in high training costs, low efficiency and even risks. Additionally, the dynamic nature of the external environment complicates the training process. For instance, during online reinforcement learning (RL), training delays and dynamic environmental changes will lead to instability in training, as the model needs to adapt to the changing environment.

OpenAI's Deep Research exemplifies the practical challenges in agent model development. As officially claimed OpenAI (2025b), Deep Research's agent model is built by end-to-end reinforcement fine-tuning atop the flagship reasoning model *o3*. The main action here is web search, and the environment consists of publicly available information on the internet.

Regarding Challenge 1, the model needs to autonomously determine during the reasoning process when its internal knowledge is insufficient, and when it needs to obtain information from the external environment. This requires the model to have a good understanding of its knowledge limitations and the ability to trigger actions at the right time.

As for Challenge 2, the cost of calling search engines and the dynamic nature of the network environment pose significant difficulties. The time and resources required for search calls can lead to high training costs and low efficiency. Moreover, the dynamic nature of the network environment makes it challenging for the model to adapt and respond in a timely manner.

In the next Section, we will use an open-domain QA task similar to Deep Research as the testbed, with the tool restricted to web search, to explore possible solutions.

3 LEARN TO GENERATE CHAIN-OF-ACTION

3.1 OVERVIEW

We introduce an agent model training solution, Automatic generation of Chain-of-Action (**AutoCoA**), which consists of two stages: Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL), to internalize the Chain-of-Action (CoA) generation capability into reasoning models.

To address the challenge of reasoning-action balance, we mix pure CoT data (w/o tool usage) in both SFT and RL stages. This ensures that the model does not forget its reasoning capabilities and can adaptively decide whether to use action on tasks of different difficulties. Meanwhile, at the beginning of SFT, we add additional substage of injecting the tool-usage ability at step level, referred to as *CoT+A*. Specifically, we generate contrastive data pairs by modifying a reasoning prefix in erroneous samples: one variant enables tool usage while the other does not. Successful task completion with tool invocation serves as a positive example, whereas failure without action forms a negative counterpart. Contrastive loss is then used to train the model on when to trigger actions.

To address the challenge of managing environment interaction, we explore the possibility of enabling the agent model to simulate the environment by incorporating an internal world model. Specifically, at the end of SFT, we added a sub-stage to train the policy model to simulate tool calls and generate the corresponding observations. During the RL stage, the model first generates its own observations based on the policy model and explores potential trajectories through extensive rollouts for fast convergence. Then, only a limited number of tool calls are made to interact with the real environment, ensuring adaptation to real-world conditions.

The overall AutoCoA framework consists of the following stages:

SFT Stage

- **CoT + A**: focusing on step-level action triggered by interleaving action tokens ⟨*action*⟩ within CoT sequences at critical reasoning junctures. Using contrastive loss, trajectories

with necessary actions (e.g., invoking a search to resolve knowledge gaps) are compared against those without, teaching the model to autonomously decide *when to action*.

- **CoT + CoA (w/ observation mask)**: the main training phase at the trajectory-level. During loss calculation for the policy model, the environment response is excluded. This teaches the model *how to action*, including determining the action type and parameters.

- **CoT + CoA (w/o observation mask)**: training the model to predict environment responses (e.g., simulated search results) alongside actions. Joint optimization of action generation and observation prediction equips the policy with an implicit world model. This preconditions the model for the subsequent RL stage.

RL Stage

- **CoT + CoA (simulated environment)**: after rolling out an ⟨*action*⟩, the policy model itself continues to generate the observation leveraging the learned internal world model from SFT. This allows for extensive sampling to explore diverse action sequences, and rapid converging to a feasible solution space.

- **CoT + CoA (real environment)**: leveraging the improved exploration capabilities from the previous sub-stage, the model interacts with the real environment to receive observations. This substage helps the model adapt to the real-world environment and ensures robust deployment.

The AutoCoA framework enables progressive capability injection—from step-level action triggering to trajectory-level CoA optimizing, and from simulated exploration to real environment interaction. In the following subsections, we introduce in detail the implementation of each sub-stage.

3.2 SUPERVISED FINE-TUNING

3.2.1 DATA SYNTHESIS

In this section, we use the training set of HotpotQA Yang et al. (2018) as our base dataset. HotpotQA is a multi-hop question-answering dataset containing questions and answers that require multi-step reasoning. We randomly selected 20,000 samples to construct our dataset $\mathcal{D}$. Formally, each data point $d \in \mathcal{D}$ contains the following elements:

- Question q: a natural language question requiring multi-step reasoning;

- Answer a: the correct answer to the question.

To support the training requirements of the AutoCoA framework in the Supervised Fine-Tuning (SFT) stage, we designed three data synthesis methods to generate CoT data, CoA data, and CoT+A data, respectively. We used the DeepSeek-R1-Distill-Qwen-32B Guo et al. (2025) model for data generation. Regarding the environment for tool invocation, we set up a local wiki search engine based on FlashRAG Jin et al. (2024). The integration of this tool enables the model to simulate the process of obtaining additional information through external resources during data generation, thereby training the model to intelligently trigger external tool calls when facing complex reasoning tasks, further enhancing its performance.

CoT data: We randomly selected 10,000 samples from dataset $\mathcal{D}$ as the base data. Using the DeepSeek-R1-Distill-Qwen-32B model with a CoT prompt designed to elicit step-by-step reasoning, we generated pure chain-of-thought processes without permitting external tool usage. The resulting CoT data captures the model's sequential reasoning, encompassing problem decomposition, analysis of known information, and derivation of the final answer, all independent of external tools. These data are designed to maintain the model's basic reasoning abilities, ensuring it can handle questions that do not require external knowledge supplementation. Formally, we define the CoT data set as $\mathcal{D}_{CoT} = \{(q, r_{CoT}, a)|(q, a) \in \mathcal{D}\}$, where r_{CoT} denotes the sequence of reasoning steps generated by the model.

CoA data: We selected another 10,000 samples from dataset $\mathcal{D}$ for construction. Using a CoA prompt that instructs the model to invoke search tools when encountering knowledge gaps, we

enabled tool-assisted reasoning. The model triggers the $\langle action \rangle$ token to call a search tool, retrieves necessary information, and resumes reasoning based on the results. The generated CoA data captures the full chain: the question q, the reasoning process r_{CoA} (including the action trigger $\langle action \rangle$, observation o from the environment, and subsequent reasoning), and the final answer a. This data structure enables the model to learn when and how to trigger actions, as well as how to continue reasoning based on the returned observations. Formally, we define the set of CoA data as $\mathcal{D}_{CoA} = \{(q, r_{CoA}, a)|(q, a) \in \mathcal{D}\}$, where r_{CoA} represents the interleaved sequence of reasoning, action, observation, and further reasoning.

CoT+A data: The CoT+A sample construction comprises two pairs of positive and negative examples. From the CoT and CoA data, we identified samples where either method yielded incorrect answers. For these, we generated contrastive pairs by altering the reasoning trajectory at a specific point: either inserting an action trigger $\langle action \rangle$ to enable tool usage or removing it to enforce internal reasoning. Each pair shares the same initial prefix but diverges on whether tool usage is permitted thereafter, leading to scenarios where either *CoT is incorrect but CoA is correct* or *CoA is incorrect but CoT is correct*. The former scenario highlights cases in which the model must leverage external tools to arrive at the correct solution, whereas the latter shows that indiscriminate tool usage can result in errors. We train these contrastive pairs using contrastive loss functions to help the model learn when to trigger actions and when to rely solely on its internal reasoning.

For the training process, we initially selected 1,500 CoT+A contrastive data points as the training set for SFT Stage 1, thereby injecting the capability to determine when to trigger $\langle action \rangle$. Subsequently, for SFT Stages 2 and 3, we combined 5,000 CoA data points with 1,000 CoT data points to instill the capability of how to execute $\langle action \rangle$. Including 1,000 CoT samples ensures the model retains its ability to solve tasks without external tools, preventing over-reliance on actions and balancing reasoning with tool usage. This mixed strategy ensures that the model can adaptively decide whether to utilize external tools based on task complexity and knowledge requirements, thereby maintaining stable performance across diverse types of questions.

3.2.2 SFT-STAGE1: COT+A

In this subsection, we describe our training approach for teaching reasoning models to decide when to switch from internal reasoning to external action—a process that can be framed as contrastive learning over sequence patterns. During this learning process, the policy model π_θ distinguishes between two distinctly different sequence output modes: one mode is initiated by the thinking-type token $\langle think \rangle$ and ultimately ends with an action-type token $\langle action \rangle$, triggering interaction with the external environment; the other mode includes only the model's internal reasoning behaviors initiated by the $\langle think \rangle$ token. Both modes share the same prefix context.

We denote by $x_{chosen} = [x_1, x_2, \ldots, x_n]$ the sequences for which we intend to boost the probability under the policy model, and by $x_{rejected}$ the sequences whose probabilities we aim to reduce. In most cases, we consider sequences containing the $\langle action \rangle$ token as x_{chosen}. However, to avoid reinforcing model biases and to help the model correctly learn when to act, we also include some sequences containing only the $\langle think \rangle$ token and ultimately arriving at the correct answer as part of x_{chosen}.

Given a context $\mathbf{c}$, we compute the log logprobability of each target output sequence. After normalization, the result can be calculated as:

$$P(x \mid \mathbf{c}) = \frac{1}{|x|} \sum_{i=1}^{|x|} \log \pi_\theta(x_i \mid \mathbf{c}, x_{<i})$$

where $|x|$ denotes the length of the sequence, and $\pi_\theta(\cdot)$ represents the probability output by the current model for the given sequence. The contrastive loss function used at this stage is thus formulated as:

$$L_{contra} = -\log \sigma \left[P(x_{chosen} \mid \mathbf{c}) - P(x_{rejected} \mid \mathbf{c}) \right]$$

where σ is the sigmoid function. This loss encourages the model to assign higher probabilities to chosen sequences than to rejected sequences.

During training, we observed a phenomenon: both chosen and rejected sequence probabilities tend to decrease simultaneously, potentially leading to model collapse. To prevent this, we incorporate an auxiliary supervised fine-tuning loss:

$$L_{aux} = -P(x_{\text{chosen}} \mid \mathbf{c})$$

This auxiliary loss encourages the model to maintain high probabilities for chosen sequences while differentiating them from rejected sequences. The final loss combines these components:

$$L = L_{contra} + \alpha \cdot L_{aux}$$

where α is a coefficient controlling the contribution of the auxiliary loss.

3.2.3 SFT-STAGE2: CoT + CoA (W/ OBSERVATION MASK)

After the previous stage, the policy model π_θ has learned to decide at the individual step level when to trigger actions for external assistance; however, it has not yet captured complete decision-making trajectories. Therefore, at this stage,we further train the model via supervised fine-tuning on synthetic complete decision-making trajectories that integrate both CoT and CoA components. This enhances the model's capability to couple multiple reasoning and action steps, teaching it a behavioral paradigm of alternating reasoning with action, while preserving its original reasoning capabilities as much as possible. Additionally, to avoid interference from external feedback during learning, we mask out loss contributions from tokens representing external feedback.

Given the task context $\mathbf{tc}$ and the complete decision-making trajectory $x = [x_1, x_2, \ldots, x_n]$, where each $x_i \in \{\langle\text{think}\rangle, \langle\text{action}\rangle, \langle\text{answer}\rangle\}$, the loss function at this stage is computed as follows:

$$L = \frac{1}{|x|} \sum_{i=1}^{|x|} I_{x_i \neq \mathbf{o}} \left[\log \pi_\theta(x_i \mid \mathbf{tc}, x_{<i}) \right]$$

Here, $I_{x_i \neq \mathbf{o}}$ indicates tokens that do not correspond to external feedback, and we only consider the numerical terms associated with these tokens in the loss calculation.

3.2.4 SFT-STAGE3: CoT + CoA (W/O OBSERVATION MASK)

In this stage, we modify the training process by including the loss for external environment feedback tokens in the complete trajectories. During training, the model is tasked with optimizing both its internal reasoning and action outputs while also accurately predicting external feedback. This joint training effectively serves as implicit learning of an internal world model, thereby offering a more robust initialization for the subsequent reinforcement learning stage.

3.3 REINFORCEMENT LEARNING

The reinforcement learning stage in the AutoCoA framework aims to internalize the Chain-of-Action generation capability into the reasoning model, enhancing its multi-turn tool usage capacity. Building on the Supervised Fine-Tuning stage, RL employs Group Relative Policy Optimization (GRPO) to refine the model's ability to interleave CoT and CoA sequences. The approach utilizes a precise matching rule reward system, ensuring that the model not only reaches the correct answer but also adheres to predefined formatting requirements.

3.3.1 REWARD FUNCTION

The reward function in RL is strictly based on two criteria: exact match with the ground truth and adherence to the required format. The model receives rewards only when the generated output exactly matches the expected result. Additionally, a format penalty is introduced to enforce the correct structure of the output sequence.

- **Exact matching reward**: A positive reward is awarded only when the final output is an exact match with the ground truth. This approach encourages the model to maintain high precision in generating responses.
- **Format penalty**: A penalty is imposed if the model's output does not adhere to the required format, such as missing critical tags like $\langle\text{think}\rangle$. This mechanism ensures the model generates well-structured CoT and CoA sequences.

3.3.2 Policy Optimization using GRPO

The model is optimized using the Group Relative Policy Optimization (GRPO) algorithm, which is adapted to the generative nature of CoT and CoA sequences. GRPO evaluates a group of potential actions at each step and optimizes the policy based on the relative performance of these actions within the group.

For each state, GRPO samples a set of actions and computes the group-relative advantage:

$$A_t = r_t - \frac{1}{K} \sum_{k=1}^{K} r_{t,k}$$

Where r_t is the reward of the selected action, and K is the size of the action group. This advantage function guides the model to prefer actions that outperform others in the same context.

The objective function for GRPO is:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \log \pi_\theta(a_t|s_t) A_t \right]$$

This method ensures that policy updates are driven by the relative quality of actions, promoting stability and efficiency during training.

3.3.3 RL-stage1: CoT + CoA (simulated environment)

In the first RL stage, the model interacts with a **simulated environment** using its internal world model developed during the SFT phase. This simulated environment acts as a controlled, low-cost, and highly efficient sandbox where the model can freely explore a diverse range of action strategies.

During this stage, the model generates hypothetical tool interactions and simulates the responses using its learned internal world model. GRPO is utilized to explore a wide array of potential CoA strategies through group-based sampling and rule-based rewards. Through iterative refinement, the model learns to balance reasoning and action, improving its ability to decide when and how to invoke external tools.

3.3.4 RL-stage2: CoT + CoA (real environment)

In the second RL stage, the agent model transitions to the **real environment**, where it interacts with actual tools to obtain real-world feedback. This phase focuses on adapting the strategies initially learned from the simulated environment to dynamic and unpredictable real-world scenarios.

Compared to the previous stage, in this stage, the model executes real tool invocations (e.g., web searches) and replaces the simulated responses with actual feedback obtained from these tools. This staged approach enables the agent model to maintain its internalized CoA generation capability, ensuring strong performance in both controlled and open-ended task environments.

4 Experiments

4.1 Settings

The evaluations in this experiment encompass a range of open-domain QA tasks, including 2 single-hop QA datasets and 3 multi-hop QA datasets.

Single-hop QA datasets:
- *Natural Questions (NQ)* Kwiatkowski et al. (2019) consists of real-world Google search queries with answers sourced from Wikipedia articles, providing a benchmark for answering straightforward questions.

- *TriviaQA* Joshi et al. (2017) is a large-scale dataset derived from trivia websites and competitions, characterized by complex entity relationships and challenging question formulations.

Multi-hop QA datasets:
- *HotpotQA* Yang et al. (2018) is the pioneering large-scale dataset that requires reasoning across multiple Wikipedia paragraphs to derive answers.
- *2WikiMultihopQA (2WIKI)* Ho et al. (2020) enhances multi-hop question answering by offering explicit reasoning paths.
- *MuSiQue* Trivedi et al. (2022) presents 2-4 hop questions constructed by combining elements from five existing single-hop datasets, testing the ability to integrate multiple information sources.
- *Bamboogle* Press et al. (2022) introduces a set of complex questions that major search engines, including Google, often answer incorrectly, offering a robust evaluation of models' compositional reasoning across diverse domains.

In this experiment, we use two evaluation metrics:
- **Exact Match accuracy (EM)** : Counts as correct only when the model's prediction is exactly the same as the ground truth. This metric strictly evaluates the model's precision.
- **LLM-based accuracy (LLM)**: Uses LLM (Qwen2.5-14B-instruct) to determine if the model's prediction semantically matches the ground truth, allowing for more diverse correct answers.

We selected a small-scale reasoning model, R1-Distill-Qwen-7B, for AutoCoA training. For parameter optimization, we utilized LLaMA-Factory Zheng et al. (2024) as the framework for Supervised Fine-Tuning and verl Sheng et al. (2024) as the framework for Reinforcement Learning. All RL training was conducted on a single node with Nvidia H20 GPUs, utilizing sequence parallelism.

4.2 COMPARED METHODS

We conducted a comprehensive evaluation of our AutoCoA framework by comparing it with various baseline methods across different categories. This section provides a brief introduction to these methods.

Model only (w/o action)

To establish a baseline for the inherent knowledge capacity of foundation models without retrieval augmentation, we evaluate several reasoning-focused models in a zero-shot setting:
- *R1-Distill-Qwen-7B* : the initial policy model used for AutoCoA training.
- *Deepseek-R1* : The original 670-B large reasoning model with enhanced reasoning capabilities, representing state-of-the-art performance without retrieval augmentation.

Agent workflow

We implemented traditional agent workflows (ReAct) on different types of base models to evaluate the effectiveness of action frameworks across model categories:
- *Qwen-7B-Base(ReAct)* : A base language model configured with ReAct prompting techniques to generate reasoning and action sequences. This represents a standard baseline for agentic workflows without specialized reasoning capabilities.
- *R1-Distill-Qwen-7B(ReAct)* : A reasoning-enhanced model configured with ReAct prompting. This approach combines enhanced reasoning capabilities with a structured action framework.

Both approaches follow standard ReAct protocols where external tool calls are explicitly structured through prompting rather than being internalized as model capabilities.

Agent model

We evaluated several variants of our agent model to understand the impact of each training stage:

- *SFT-stage1* : The model after completing only the CoT+A stage, which focuses on teaching when to $\langle action \rangle$ using contrastive learning at the step level. This stage used 1.5K CoT+A contrastive data points.

- *SFT-stage2* : The model after completing only the CoT+CoA(w/ observation mask) stage, which focuses on trajectory-level action patterns without observation mask prediction. This stage used a mix of 5K CoA and 1K CoT data points.

- *SFT-stage1&2* : A combined approach where the model undergoes both SFT-Stage 1 and SFT-Stage 2 sequentially, benefiting from both step-level action triggering and trajectory-level pattern learning.

- *SFT-stage1&2&3* : The complete SFT pipeline, including the CoT+CoA(w/o observation mask) stage that enables the model to predict environment responses alongside actions, effectively learning an implicit world model. This stage used the same data mix as SFT-stage2.

We compared the performance of different RL settings and methods. All training was conducted with 96 optimization steps on the same 4,608 problems:

- *RL-stage2*: In this setting, we utilize in-context learning to guide a model without SFT through CoA-based reasoning. An additional 50% optimization steps were added for this training configuration.

- *SFT-stage1&2&3 + RL-stage1*: The model is trained exclusively in the simulated environment, using *SFT-stage1&2&3* as the initial policy model.

- *SFT-stage1&2&3 + RL-stage1&2*: The model is first trained for the initial 80 steps (5/6) in the simulated environment, then switched to the real environment for the rest 1/6. The initial policy model is set to *SFT-stage1&2&3*.

- *SFT-stage1&2 +RL-stage2*: The model is trained exclusively in the real environment, using *SFT-stage1&2* as the initial policy model.

4.3 RESULTS

The main results are summarized in Table 1. Observations include:

- *Agentic workflow vs. agent model*. After CoA learning, all AutoCoA variants substantially outperform the initial policy model with the ReAct workflow. This validates the effectiveness of end-to-end optimization for task-oriented when and how to act during reasoning.

- *Agent model: SFT-stage1&2 vs. SFT-stage2*. Separating CoA fine-tuning into *when-to-act* (stage1) and *how-to-act* (stage2) sub-stages enhances the model's ability to learn CoA patterns, enabling better transitions between thought and action.

- *Agent model: RL-stage2*: Compared to the initial policy, direct RL shows some improvement but is constrained by the initial policy's baseline capacity, resulting in marginal gains. Direct RL application shows marginal improvements over the initial policy, constrained by the policy's baseline capacity.

- *Agent model: SFT-stage1&2&3 + RL-stage1 vs. SFT-stage1&2&3 +RL-stage1&2*. After extensive (5/6) simulated environment training (RL-stage1), a small portion (1/6) of real-world interaction (RL-stage2) improves adaptation to dynamic environments, further boosting accuracy.

- *Agent model: SFT-stage1&2 + RL-stage2 vs. SFT-stage1&2&3 +RL-stage1&2*. Compared to full real-world interaction, initial simulated environment training sacrifices some performance. However, it significantly reduces interaction costs. Achieving acceptable results with only 1/6 real interactions demonstrates the feasibility of internal world modeling. Future work is needed to improve world model fidelity and leverage the saved costs for advanced training paradigms.

Table 1: Performance comparison between different QA methods. **Bold** indicates the highest value, while <u>underline</u> indicates the highest value among the different settings of *AutoCoA*.

Method	Single-hop QA				Multi-hop QA								Average	
	NQ		TriviaQA		HotpotQA		2WIKI		MuSiQue		Bamboogle			
	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM	EM	LLM
Model only (w/o action)														
R1-Distill-Qwen-7B	5.9	9.8	17.3	21.1	11.7	14.1	23.3	25.9	1.1	2.5	11.7	13.1	11.8	14.4
Deepseek-R1*	**39.0**	**54.2**	**78.2**	**85.6**	**44.8**	**53.0**	**52.4**	**55.8**	**21.2**	**27.4**	**56.8**	**63.2**	**49.5**	**60.3**
Agentic workflow														
Qwen-7B-Base (*ReAct*)	18.0	37.8	35.2	48.8	17.2	28.4	24.0	30.8	4.2	8.4	19.2	24.8	19.6	29.8
R1-Distill-Qwen-7B (*ReAct*)	15.1	20.9	25.7	30.1	15.1	18.6	21.3	22.9	1.9	3.9	12.3	14.7	15.2	18.5
Agent model: SFT														
SFT-stage1	22.7	32.2	40.0	47.8	21.2	28.3	26.4	29.6	4.8	8.2	20.2	22.8	22.7	29.1
SFT-stage2	25.0	33.1	42.6	48.2	31.7	39.2	43.8	48.4	8.5	12.1	24.3	27.2	29.5	35.6
SFT-stage1&2	25.0	34.3	45.6	51.3	33.0	40.8	48.1	52.4	9.0	12.4	<u>30.4</u>	<u>33.3</u>	32.0	38.5
SFT-stage1&2&3	23.8	33.1	44.7	51.2	30.1	36.5	49.2	52.0	10.5	14.7	28.8	32.0	31.3	37.7
Agent model: RL														
RL-stage2	22.8	30.0	37.5	43.2	23.2	28.2	33.2	33.8	5.6	8.2	20.0	23.2	23.7	27.8
SFT-stage1&2&3+RL-stage1	26.0	33.4	47.2	53.8	31.4	40.0	48.6	51.4	9.6	12.4	27.2	32.0	31.7	37.2
SFT-stage1&2&3+RL-stage1&2	<u>28.6</u>	<u>37.0</u>	48.6	54.0	33.2	41.2	<u>49.6</u>	<u>53.6</u>	<u>12.6</u>	<u>16.2</u>	28.0	31.2	33.4	<u>38.9</u>
SFT-stage1&2+RL-stage2	28.4	34.6	<u>52.4</u>	<u>56.6</u>	<u>35.6</u>	<u>42.8</u>	48.4	51.2	12.0	15.2	26.4	30.4	<u>33.9</u>	38.5

* The results reflect R1's knowledge memorization rather than action capabilities, due to its large parameter size and the outdated knowledge in QA questions. On new questions, R1's accuracy is very low. See Appendix.A for details.
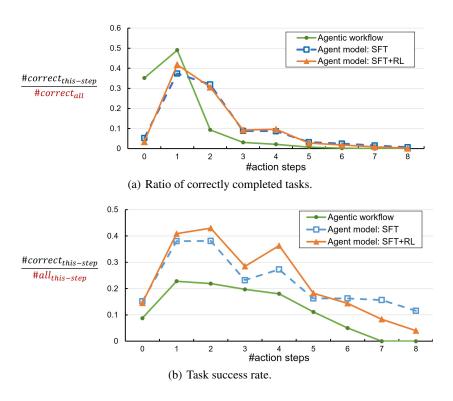
(a) Ratio of correctly completed tasks.

(b) Task success rate.

Figure 3: Analysis of hong-horizon execution capabilities.

4.3.1 IMPROVING LONG-HORIZON TASK EXECUTION

To evaluate the role of CoA learning in tackling tasks that require long-horizon reasoning and action, we analyzed model performance before and after CoA learning as the number of executed actions ($\#action$) increased.

Fig. 3(a) illustrates the distribution of correctly completed tasks across different $\#action$. Without CoA learning, the initial policy model primarily succeeded on tasks requiring only 1 or 2 actions. After CoA learning, agent models demonstrated the ability to support longer thought/action rounds, enabling them to complete more complex tasks.

We further examined task success rates by $\#action$ (Fig. 3(b)). The results reveal a consistent trend with Fig. 3(a): Initial policy with workflow shows declining success rates as $\#action$ increases, whereas agent models maintain a relatively high accuracy at 5 actions. This demonstrates that task-oriented end-to-end training enables agent models to learn improved interleaving patterns between thought and action, with strong consistency in long-context scenarios as the number of actions increases.

5 CONCLUSION AND FUTURE WORK

We propose AutoCoA, an agent model learning approach to enhance the multi-turn tool use capability on top of reasoning models. Through end-to-end fine-tuning, the derived agent model demonstrates a superior ability to determine when and how to act in pursuit of the ultimate goal, outperforming traditional agentic workflows in goal-oriented decision-making.

Methodologically, this report represents an initial exploration into learning CoA (Chain-of-Action). It serves as a preliminary attempt, with a wide range of potential approaches yet to be explored, including pure reinforcement learning (RL), improved action-inclusive sampling, and loss design for trajectories that interleave CoT (Chain-of-Thought) and CoA.

From a task perspective, the next steps are extending the application scope: (1) Agent model for open-ended tasks, exploring methods for implementing agent models capable of handling open-ended tasks with rubrics for grading; (2) AgentRFT (Reinforcement Fine-Tuning), building on our prior work with OpenRFT Zhang et al. (2024b) to tailor the agent foundation model to meet specific task objectives and leverage specialized tools within particular scenarios.

5.1 AGENT ROADMAP

OpenAI has long pursued the vision of seamless environmental interaction. *Operator* stems from its early key project "Computer Use", and *Deep Research* continues its goal since "Plugins" and "GPTs". However, the earlier attempts were constrained by the limited capabilities of models at the time.

Today, with advancements in language modeling and logical reasoning, Deep Research has tackled highly complex tasks using only basic tools like web search and document parsing. This demonstrates a universal recipe for agent development: once a model's reasoning capacity surpasses a certain threshold, providing it with a few simple tools and a verifiable task objective within a controlled environment enables what the Claude 3.7 official blog describes as "action scaling". The fact that OpenAI has began to assess model capabilities via replaceable human expert hours is particularly noteworthy. Even if a model only possesses an average level of human intelligence and tool usage ability, combined with the efficiency of AI in information acquisition and processing, the enhancement to productivity would be explosive.

Operator and Deep Research exemplify OpenAI's agent paradigm—combining execution (Operator as the "hands") and reasoning (Deep Research as the "brain") to achieve dynamic programming, autonomous tool usage, asynchronous processing, and automatic task completion. We categorize their functionalities as *Computer Use* and *Knowledge Research*, respectively, and envision a three-phase roadmap for agent development (Fig. 4).

5.1.1 SHORT-TERM: APPLICATIONS

By enhancing reasoning foundation models with basic tool-usage capabilities to create agent foundation models, practical applications are enabled through methods like Reinforcement Fine-Tuning (RFT). Below are the key scenario applications for this stage:

Agentic RPA
A natural evolution of traditional Robotic Process Automation (RPA), Agentic RPA integrates learning and reasoning to automate enterprise workflows (e.g., ERP/CRM systems). Unlike rule-based RPA, which struggles with unstructured data or dynamic environments, Agentic RPA adapts to evolving business needs, can tackle more complex and flexible tasks.

Personal assistant
Focused on mobile-centric, personalized interactions, this agent leverages behavioral logs and personal knowledge bases to act as a user's "digital twin." Future extensions include proactive behavior (predicting needs rather than passive responses) and multimodal context awareness (perceiving and interpreting environmental cues to refine task execution).

Domain analyst
Deep Research currently faces two main limitations: restricted open-domain data access and shallow expert analysis. For specialized fields such as law, medicine, and research, addressing these gaps requires fine-tuning on proprietary data and domain knowledge. This may enable the leap into actionable insights, supporting real-world analytical consulting and decision-making.

Coding agent
In addition to applying computer use and knowledge research capabilities, another critical short-term direction is automating software engineering, as exemplified by projects like *Devin* and *Manus*. Coding agents treat programming not as an end goal but as a tool-use mechanism (e.g., automating API integrations). Whether deployed internally or as standalone products, coding agents represent foundational infrastructure for scaling agent capabilities.
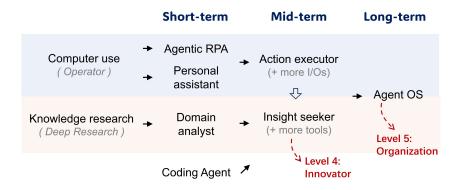
Figure 4: Agent roadmap.

5.1.2 Mid-term: Expanding Capabilities

Agent evolution in the mid-term will focus on two parallel directions: (1) Action executor, enhancing task execution by accessing a broader range of I/O interfaces, evolving into "more capable hands and feet." (2) Insight seeker, deepening information processing through expanded tool integration, becoming a "more comprehensive brain."

Action executor

Beyond integrating more software application APIs, agent task execution will extend from software operations to the physical world, requiring enhanced interfaces for both input perception and output execution. IoT device interfaces are typical inputs. By connecting to IoT devices in smart homes and factories, Agent can better sense the environment, and understand human behavior and needs for automated task execution. Output execution interfaces will involve remote operation and control for physical-world automation.

Insight seeker

At the Deep Research release, OpenAI noted that search is just the first step, and hopes to integrate more tools in the future to let the Agent "autonomously discover new knowledge". Possible tools include information acquisition (data sources, unstructured information parsing) and information processing (general visualization, database analysis, and domain-specific tools).

Currently, knowledge research agents mainly focus on discovery and information integration within the language space. Innovation and insight require more input. The current input is provided by human feedback in the human-machine collaboration paradigm. In the future, inputs could come from new findings through tool experiments or feedback from multi-modal intelligence.

At that time, information processing will no longer start with explicit user requests nor take the form of simple Q&A. Instead, the role will shift: we will provide agents with data, experimental materials, and simulation environments, allowing them to autonomously explore, analyze, deduce, and generate new insights. With the continued development of action executors and coding agents, the insight seeker could further evolve toward the fourth stage: **Innovator**.

5.1.3 Long-term: Agent OS

Web pages and apps have served as the application carriers for internet and mobile internet technologies, respectively, while agents are now emerging as the carriers for artificial intelligence technologies. From a computational perspective, agent models function as the CPU in traditional computer architecture, handling core computation and task orchestration. Tools function as software programs, executing specific operations, and short-term and long-term memory correspond to RAM and hard drives. Together, these components form the basis of what could be termed an *agent OS*.

However, building an agent OS presents significant challenges. Much like the von Neumann architecture relies on the unified storage of programs and data alongside a complete instruction set, agent OS requires unified representations for multimodal I/O, tools, and memory, as well as an instruction

set tailored for agents. This poses difficulties for current Transformer models, which depend on token-based sequential representations, and for natural language-based instruction systems.

Furthermore, data representation and organizational structures need to be adapted to suit agent models. GUIs like web pages and apps, designed for human interaction, are not the most efficient for agents. Agent protocol is required to define how agents interact with tools, vector databases, and other agents, enabling operations at a more foundational level to fully harness their strengths.

As the agent ecosystem matures, multi-agent collaboration will become pivotal. Agents will dynamically coordinate in complex projects–sharing intermediate results, delegating subtasks, and self-organizing like a human team. This evolution from individual competence to collective intelligence, marked by enhanced autonomy and cooperative problem-solving, may signal the emergence of the fifth stage: **Organization**.

REFERENCES

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jie He, Nan Hu, Wanqiu Long, Jiaoyan Chen, and Jeff Z Pan. Mintqa: A multi-hop question answering benchmark for evaluating llms on new and tail knowledge. *arXiv preprint arXiv:2412.17032*, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*, 2024.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

OpenAI. Gpt-4 technical report. Technical report, OpenAI, 2023. URL `https://cdn.openai.com/papers/gpt-4.pdf`.

OpenAI. Openai o1 system card. Technical report, OpenAI, 2024. URL `https://cdn.openai.com/o1-system-card-20241205.pdf`.

OpenAI. Introducing operator, January 2025a. URL `https://openai.com/index/introducing-operator/`.

OpenAI. Introducing deep research, February 2025b. URL `https://openai.com/index/introducing-deep-research/`.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 8634–8652, 2023.

Hugging Face Team. Open-source deepresearch – freeing our search agents. *Hugging Face Blog*, 2025. URL `https://huggingface.co/blog/open-deep-research`.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Jason Wei, Barret Zoph, Sainbayar Sukhbaatar, Jaime Pineda, Tom Hashimoto, Henry Ma, Quoc Le, and David Dohan. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 15208–15221, 2022.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, and Karthik Narasimhan. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL `https://arxiv.org/abs/2210.03629`.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. *arXiv preprint arXiv:2412.14135*, 2024.

Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. o1-coder: an o1 replication for coding, 2024a.

Yuxiang Zhang, Yuqi Yang, Jiangming Shu, Yuhang Wang, Jinlin Xiao, and Jitao Sang. Openrft: Adapting reasoning foundation model for domain-specific tasks via reinforcement fine-tuning, 2024b.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL `http://arxiv.org/abs/2403.13372`.

A APPENDIX

Table 2: Performance Comparison across Temporal Knowledge Domains

Dataset	Recent (*Unknown*)				Popular& Old (*Known*)			
	Main Question		Sub-question		Main Question		Sub-question	
Model	EM	LLM	EM	LLM	EM	LLM	EM	LLM
DeepSeek-R1-Distill-Qwen-7B	0.026	0.052	0.017	0.039	0.052	0.073	0.231	0.267
DeepSeek-R1	0.078	0.098	0.079	0.119	0.615	0.646	0.899	0.944

A.1 DATASETS

This work evaluates how model knowledge memorization impacts multi-hop reasoning accuracy. The experiments are structured as follows:

1. **Unknown knowledge evaluation**: A test set of 192 three-hop questions (576 sub-questions) was constructed using time-sensitive corpora sourced from 97 sports-related reports published by CCTV News between January 16, 2025, and March 7, 2025. This dataset ensures that the models were not exposed to the data during training, allowing for an evaluation of their ability to handle unseen temporal knowledge.

2. **Known knowledge evaluation**: For comparison, we used the MINTQA-POP benchmark dataset, proposed by He et al. (2024). We use a subset of 192 three-hop popular questions (576 sub-questions) was selected through stratified sampling. This dataset covers high-frequency knowledge domains familiar to the models from wiki datasets, enabling an assessment of model performance on known knowledge.

A.2 EXPERIMENTAL RESULTS

The results highlight the models' distinct performance across the two evaluation scenarios:

1. **Performance on questions requiring known knowledge**: R1 achieved *61.5%* (EM) and *64.6%* (LLM) on MINTQA-POP, vastly outperforming the distilled 7B model's *5.2%* (EM) and *7.3%* (LLM). This demonstrates R1's reliance on memorized patterns rather than generalized reasoning–it excels on tasks involving familiar knowledge domains.

2. **Performance on questions requiring unknown knowledge**: Both models collapsed on unseen temporal data: R1 dropped to *7.8%* (EM) and *9.8%* (LLM), and R1-Distill-Qwen-7B dropped to *2.6%* and *5.2%* (LLM). The drastic performance drop highlights a fundamental knowledge cutoff dependence. As noted in Section 4.3, R1's superior performance stems from memorizing outdated wiki facts–when tested on temporal questions requiring very recent knowledge, its accuracy declines significantly, underscoring the necessity of augmenting internal knowledge through tool usage from external environments.