Federation over Text: Insight Sharing for Multi-Agent Reasoning

Dixi Yao
University of Chicago
Chicago, IL, USA
dixi.yao@uchicago.edu

Tahseen Rabbani
University of Chicago
Chicago, IL, USA
trabbani@uchicago.edu

Tian Li
University of Chicago
Chicago, IL, USA
litian@uchicago.edu

Abstract

We propose a federated learning-like framework, *Federation over Text* (FoT), that enables multiple clients solving different tasks to collectively generate a shared library of metacognitive insights by iteratively federating their local reasoning processes without sharing actual problem instances or task instructions. Instead of federation over gradients (e.g., as in distributed training), FoT operates at the **semantic level** without any gradient optimization or supervision signal. Iteratively, each client runs an LLM agent that does local thinking and self-improvement on their specific tasks independently, and shares reasoning traces with a central server, which aggregates and distills them into a cross-task (and cross-domain) *insight library* that existing and future agents can leverage to improve performance on related tasks. Experiments show that FoT improves reasoning effectiveness and efficiency across a wide range of challenging applications, including mathematical problem solving, cross-domain collaboration, real-world daily tasks, and machine learning research insight discovery. Specifically, it improves average performance scores by 25% while reducing the reasoning tokens by 4% across the first three applications. In the research insight discovery application, FoT is able to generate insights that cover over 80% of the major contributions in the subsequent papers.

ACM Reference Format:
Dixi Yao, Tahseen Rabbani, and Tian Li. 2026. Federation over Text: Insight Sharing for Multi-Agent Reasoning. In *Proceedings of The First Workshop on Agent Skills — Design, Evaluation, and Optimization of Procedural Knowledge for LLM Agents (AgentSkills).* ACM, New York, NY, USA, 44 pages. https://doi.org/XXXXXXX.XXXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
AgentSkills, San Jose, CA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2026/05
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

LLM-powered agents are increasingly deployed for complex, multi-step tasks [e.g., 1–6]. However, they can suffer from two major limitations. First, reasoning can be inefficient: agents often reason from scratch given an input instance, repeatedly solving similar problems, which can be computationally expensive and slow. Second, reasoning in multi-agent systems typically occurs in an isolated manner, where agents are chained by passing prompts or intermediate outputs [e.g., 7]. While agents exchange task solutions within a session, their internal thinking processes and full reasoning histories are not explicitly stored or communicated. As a result, these hard-won thinking processes are lost and cannot be reused or shared across agents.

It remains underexplored how to automatically transfer, reuse, or distill independent reasoning traces to help guide and improve future reasoning processes for similar agents or across domains. At the same time, classical distributed or federated paradigms for knowledge sharing [8] may not be suitable for learning generalizable insights from agents' reasoning traces. Averaging the gradients or weights of entire (or part of) large generative models can be computationally prohibitive, impossible for closed-source LLMs, and more importantly, not interpretable as parameter aggregation is not explicitly equivalent to sharing *abstract, high-level reasoning summaries.*

To this end, we propose *federation over text* (FoT), a federated-like framework that enables a set of agents to collaboratively and iteratively build a shared, evolving library of reusable *insights* from independent local reasoning traces (expressed in natural language) without communicating actual problem instances. Unlike traditional multi-agent systems that decompose a single complex task into subtasks, FoT operates across multiple independent agents over diverse but related tasks. Each agent solves some task with a base language model, and our goal is to generate an insight library—a high-level aggregation of the common principles and ideas shared across the agents' tasks.

Unlike traditional distributed learning, which averages gradients or model updates in a high-dimensional, uninterpretable space, FoT operates at the **semantic**, text level without any parameter optimization, fine-tuning, or reinforcement learning. In particular, at each round, agents transmit concise metacognitive summaries of their problem-solving

processes to a central server (which can be physical or logically virtual). The server, acting as an aggregator, identifies and distills recurring reasoning fragments from these summaries into new, explicit insights and sends the insight library back to the agents for the next round. We note that FoT is a *general* framework that allows each agent to use any local reasoning model combined with any local self-improvement workflows for their individual tasks, as long as summarized reasoning traces are shared at each iteration. Meanwhile, we note that our proposed FoT approach is analogous to the classic federated/distributed learning setting, where these two regimes can be interpreted within the same conceptual framework (side-by-side comparisons in Table 1). Additionally, FoT offers similar benefits as FL in that only abstracted reasoning traces are shared instead of raw problem instances. But operating in the semantic text space as in FoT provides more interpretable control on what information is transmitted, and allows us to identify and block sensitive information more easily.

We apply FoT to various applications including mathematical problem solving, multi-domain collaboration (scientific question-answering, coding, and Humanity's Last Exam), real-world daily tasks, and research insight discovery (Section 4). We compare our framework with a range of baselines, such as isolated reasoning without collaboration and other forms of cross-agent knowledge sharing. Across the first three applications with previous benchmarks, FoT improves the accuracy performance score by 25% on average while reducing the reasoning tokens by 4%. For the research insight discovery application, we demonstrate that an insight library learned from ICLR papers in one year can cover the core technical contributions of over 80% of accepted papers in the next year. FoT enables LLM-based systems to function not only as performance-driven solvers, but also as tools for decision support and insight discovery.

2 Related Work

Multi-Agent Collaboration. Extensive prior multi-agent works focus on enabling agents to *solve a given complicated task* in a collaborative way. A common paradigm involves decomposing a task into smaller subtasks, assigning each to an individual agent, and aggregating the partial results [e.g., 9–13]. Other existing approaches include multi-agent debate [2] or allowing multiple agents to repeatedly perform the same task and vote on a solution [14–16]. Previous works also study large-scale agentic knowledge exchange by building concrete workflows for agent laboratories [4], or simulating research communities [5, 17]. Rather than using multiple agents to handle different components of a single task or building end-to-end workflows to simulate complicated systems, in FoT, each client runs a local agent that solves problems independently and the server aggregates reasoning traces from the problem-solving processes to distill shared insights (Figure 1).

Collaborative and Federated Learning. Collaborative or federated learning (FL) [8] aggregates model parameters from multiple clients while keeping data local. Existing literature on federated agents primarily focuses on training language models in the classical parameter or gradient-level aggregation framework [18], which can be expensive and insufficient for capturing high-level abstract reasoning strategies. Social learning [19] allows teacher agents to generate synthetic examples or abstract prompts that are aggregated to train a student model in a single-task setting. FL and FoT share interesting connections (Section 3.1); FoT operates in the semantic space while being compatible with heterogeneous agent models and task domains.

LLM Skills and Recursive Self-Improvement. To embed past experience into LLM-based agents, existing approaches such as Anthropic Skills [20], Agents.md[1] , vibe research [6], and Scientific Skills [21] focus on providing concrete skills for executing specific tasks rather than generalizable, abstract insights. Moreover, they require some extent of human-guided process to generate skills, whereas FoT generates reusable insights automatically without any human intervention; even labels indicating whether an answer is correct are not required. ExpeL [22] stores an agent's past successful trajectories along with reflections on successes and failures, relying on explicit labeling. There is also extensive research on improving agent reasoning through self-reflection, self-improvement, or context engineering [1, 23–30]. However, they do not consider jointly learning insights by distilling reasoning traces across independent agents solving *different*, but related tasks. Moreover, FoT can adopt any existing self-improvement method as the local thinking approach (Section 6.1) and aggregate their traces to iteratively form generalizable insights.

Another line of work on agent improvement requires gradient optimization via various forms of fine-tuning or reinforcement learning [31, 32]. FoT operates purely on the inference level without any fine-tuning or complicated supervision. In addition, these approaches are mainly designed for solving a single task and do not consider federation. Hence, in this paper, we only compare with methods that only involve API access without computing gradients.

3 Federation over Text

In this section, we propose our federation over text framework and discuss its connections with federated learning (i.e., gradient aggregation) [8]. Federated learning (FL) focuses on training a shared global model to serve existing and future clients, whereas FoT aims to build an insight library that helps both existing and new agents to better solve a variety of tasks. In the most common FL workflow, the

[1] https://agents.md/

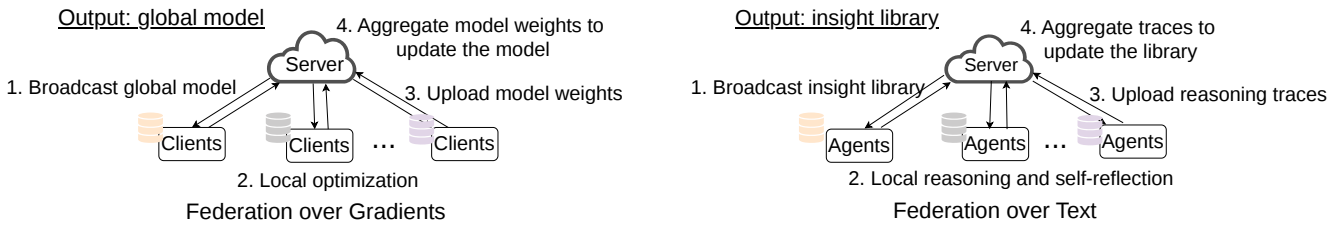

Figure 1. Workflow of federation over (accumulated) gradients and federation over text. See pseudocode in Algorithms 1 and 2 in the appendix.

Table 1. Connections between Federated Learning over Gradients and Federation over Text (FoT).

Components	Federation over Gradients	Federation over Text (FoT)
Learning Objective	Learn a global model that generalizes to test data or new clients	Learn an insight library that improves the reasoning of agents
Learning Work	Local training via *any* optimization method	Local task solving via *any* self-improvement approach
Data Minimization	Share model updates rather than raw data	Share abstracted reasoning traces rather than raw instances
Local → Server	Clients upload model weights or gradients	Agents upload reasoning traces from problem-solving
Server Aggregation	Averaging over model weights or gradients	Aggregating over reasoning traces (text) using curated prompts
Server → Local	Current learnt global model	Current curated insight library
Personalization	Maintaining personalized local models	Addressing diverse local tasks across different domains.

server broadcasts the current global model to each client, who then performs local training on their local data and transmits the local model updates back to the server. The server aggregates the model updates to form a new global model. This iterative process repeats until convergence. At a high level, in FoT, at each round, each client runs a local agent (any base LLM(s)) who solves its local task with the help of the current insight library, and sends the reasoning traces to the server. The server merges them to update the library and distributes it to all agents for the next round. We present the overall workflow of FL and FoT in Figure 1, as well as the pseudocode of both algorithms in Appendix B.

In the workflow of FoT (the right subfigure of Figure 1), each of the (potentially many) decentralized agents solves a specific problem or task, using a selected base LLM. These tasks can span different domains such as mathematical reasoning, coding, and scientific question-answering. Given a task (e.g., a collection of math questions), an agent first solves the task (e.g., generates solutions), then reflects on whether the reasoning process contains reusable techniques and what key skills can be extracted. Instead of uploading complete, raw reasoning histories, each agent generates a logic-level *reasoning trace* for each problem based on its solution and reflections, specifying when and how to apply the extracted skills via step-by-step instructions. Agents then share these shorter, summarized messages with the server, while keeping problem instances and raw histories local.

In each round, after collecting traces, the server needs to aggregate them to update the insight library. We develop one specific aggregation technique that distills high-level insights rather than simply concatenating reasoning traces. In particular, the server first clusters reasoning traces that share similar skills, and then uncovers connections among traces that solve the same problem using different skills and can potentially be merged. The entire process is entirely handled by the LLM. Specifically, the server first takes all reasoning traces as input to infer clusters and relationships among them, and then uses the resulting clusters, relationships, and original reasoning traces as input to generate the final insight library with curated prompts. Finally, the server broadcasts the updated insights library to all agents for the next iteration. The aggregated library can be re-used as an external reference to improve the reasoning of either existing agents or future agents that are not in the current network. Unlike parameter updates, the resulting knowledge is human-readable and interpretable. This improves transparency and enables cross-domain transfer of reasoning strategies. We provide all the designed prompts that we use in Appendix G.

Local Reasoning and Global Aggregation Strategies. We note that many self-improvement methods proposed by prior works [e.g., 1, 23–26] can be plugged into FoT to generate the reasoning traces sent to the server. In our experiments (Section 6.1), we plug several different local thinking

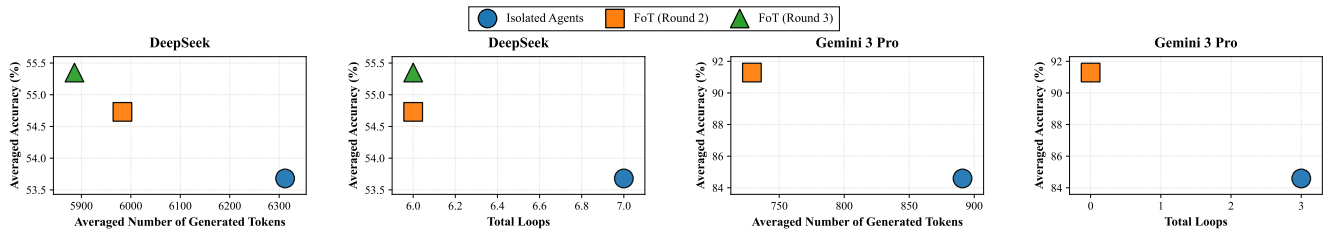

Figure 2. Comparison of reasoning accuracies and efficiency (measured by average number of reasoning tokens and total loops), averaged over eight math benchmarks.

methods into FoT. We also enable calling external tools (Section 4.3). Additionally, we note that while we currently use curated prompts to perform global aggregation in the text space (Appendix G), the aggregation step of FoT can adopt other techniques developed for general context compression [33, 34] (Section 6.2). FoT is modular in that local or global procedures can adapt various algorithms/prompts, and be handled via heterogeneous LLM models (Section 6).

3.1 Analogies Between Federated/Distributed Learning and FoT

We reiterate that there are rich connections between FL and FoT. In particular, in Table 1, we summarize how their algorithmic components can be mapped between each other. For instance, in FL, clients may adopt different optimization methods to solve local subproblems. Analogously, in FoT, each agent may leverage distinct local reasoning strategies and prompt designs to generate traces. Inspired by techniques from distributed or federated learning, FoT allows for an interesting design space to improve efficiency and effectiveness of multi-agent collaborative reasoning (Section 4). It is worth continuing to explore the design space of FoT, such as personalization strategies and handling other forms of distribution drifts across agents, which we leave for future work.

4 Applications of FoT

In this section, we study four applications demonstrating how FoT enhances reasoning performance and efficiency, including mathematical problem-solving with varying levels of difficulty (Section 4.1), cross-domain collaboration involving math, scientific QA, coding, and others (Section 4.2), real-world daily tasks (Section 4.3), and ML research insight discovery (Section 4.4). Our code and experiments are open sourced at https://github.com/dixiyao/FoT.

4.1 Mathematical Problem Solving

In this application, the agents' goal is to solve math problems. There are eight agents, each working on one of eight datasets with varying levels of difficulty, AIME24 [35], AIME25 [36], AMC, CCEE, CNMO, WLPMC, V202412 Hard, and V202505 Hard from LiveMathBench [37]. We experiment with DeepSeek-R1-Distill-Qwen-7B and Gemini-3.0-Pro as two choices of the base LLM.

Figure 2 shows the results averaged on eight benchmarks. Numbers on each benchmark are in Appendix D.1. FoT can help reduce loops (i.e., the number of times repeated sentences are generated during the reasoning process). With the presence of an insight library, agents generate fewer tokens while achieving better final performance. Additionally, FoT reduces the number of loops, indicating less repeated reasoning content. Note that our insight library size is manageable, which is helpful for communication efficiency and reducing the context length for LLMs. For DeepSeek, by Round 3, we generate 249 reasoning traces in total (33K tokens) and aggregate them into 11 insights (3K tokens). For Gemini 3.0 Pro, we generate 549 traces (53K tokens) and aggregate them into 37 insights (12K tokens). The cost of uploading reasoning traces and downloading insights is moderate, yielding substantial improvements in utilities. We study the effects of insight library size in Section 6.4.

4.2 Multi-Domain Collaboration

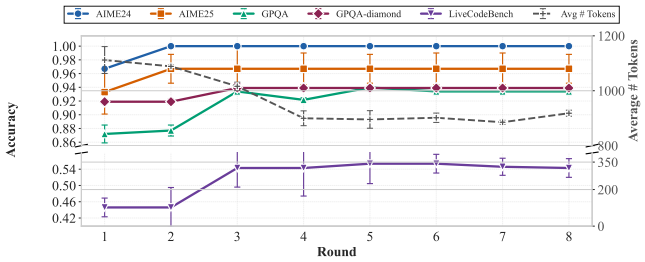

Figure 3. Accuracies and average number of generated tokens during the reasoning process across math, scientific QA, and coding benchmarks. Round 1 represents isolated agents. Insight libraries improve reasoning across domains while reducing token usage.

In this section, we go a step further to understand how FoT can retrieve cross-domain knowledge, helping agents working on different but related domain tasks collaborate and improve each other's reasoning. We *first* choose AIME24 and AIME25 for math tasks, GPQA and GPQA-diamond [39] for

Table 2. Performance scores (pass@1) and efficiency when isolated OpenClaw agents and FoT resolve all tasks in PinchBench. FoT can greatly improve accuracy and reduce the number of generation tokens. In Appendix D.3.1, we further show that the additional token cost incurred by local reasoning trace generation and insight library construction is acceptable.

LLM (Thinking Level)	Algorithms	Score (out of 100)			Execution Time (s)			# Generation Tokens (K)		
		Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
Gemini 2.5 Flash Lite	Isolated Agents	27.90	20.12	12.70	**2932**	**2835**	**2157**	57	55	54
	FoT (Round 2)	**53.29**	**52.11**	**50.36**	3667	3466	3223	57	**54**	**52**
Gemini 3.1 Pro (Low)	Isolated Agents	79.97	72.67	59.57	3174	3093	2967	**46**	44	40
	FoT (Round 2)	**86.88**	76.46	65.09	**2808**	**2528**	**1940**	48	45	40
	FoT (Round 3)	86.59	**80.44**	**76.13**	3080	2797	2553	47	44	42
Gemini 3.1 Pro (Medium)	Isolated Agents	89.06	77.64	60.36	2168	1893	1714	47	41	**33**
	FoT (Round 2)	86.48	81.04	77.07	1984	1789	1664	**44**	**38**	42
	FoT (Round 3)	**89.51**	**81.85**	**77.38**	**1889**	**1711**	**1568**	45	43	42
Gemini 3.1 Pro (High)	Isolated Agents	80.50	76.29	72.21	1864	**1764**	1683	53	50	46
	FoT (Round 2)	**87.32**	84.82	81.71	**1845**	1787	**1658**	52	48	46
	FoT (Round 3)	87.14	**85.00**	**82.11**	2091	1962	1771	**48**	**47**	**44**

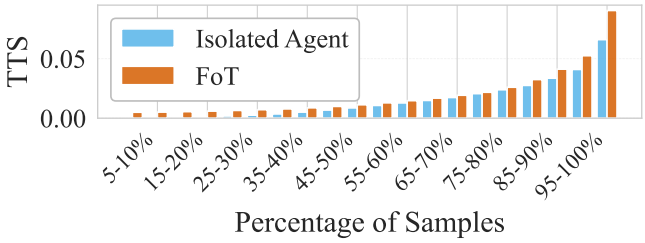

Figure 4. Comparisons of the true-thinking score (TTS) [38], measuring reasoning steps whose removal would lead the agent to a different answer. FoT exhibits fewer decorative reasoning steps.

scientific question-answering (QA) tasks, and LiveCodeBench (version 6, Hard) [40] for coding. At first glance, these three domains appear loosely related. However, apart from the performance improvement demonstrated in Figure 3, FoT generates insights that highlight underlying conceptual similarities across multiple domains. We let each agent mark which insights contribute most to the reasoning process. In Appendix D.2, we present **two case studies** illustrating the content of insights, which insights are used, and how they are applied during the reasoning process to arrive at correct solutions. These examples show that insights from one domain (e.g., math, Sudoku) can be used to correct the reasoning processes of tasks in a different domain (e.g., chemistry). *Second*, we also experiment with cross-domain collaboration on more challenging benchmarks, including Humanity's Last Exam [41], and show that even when most reasoning traces originate from incorrect answers, the insight library can still improve overall performance (Table 9). From Figure 3, we can also observe diminishing gains after round 3, suggesting convergence of the insight library.

True-Thinking Abilities. To see how FoT can also reduce decorative reasoning, we report the 'true thinking score' [38] in the multi-domain collaboration application. The true thinking score (TTS) measures the proportion of reasoning steps whose removal would lead the agent to a different answer, thus serving as a proxy metric of effective reasoning. From Figure 4, we can see that FoT increases the score compared with the baseline, indicating that insights generated by FoT can help agents skip redundant reasoning steps.

4.3 Real-World Daily Tasks

In this application, FoT positions agents as tools for decision support in daily work tasks. We use OpenClaw [42], experimenting with both Gemini 3.1 Pro and Gemini 2.5 Flash Lite as backend models. We evaluate on PinchBench [43], which covers tasks such as web research and stock analysis. Each OpenClaw agent is assigned to one task and produces reasoning traces from its execution trajectory. FoT aggregates these traces into a shared insight library, `insight.md`, which agents then use alongside pre-defined tools and skills to complete tasks. We compare with three **additional baselines** where we (1) append all agent trajectories into a retrieval-augmented generation (RAG) datastore for each agent, or (2) equip each agent with 52 pre-defined OpenClaw skills. Agents perform independent reasoning in both cases. (3) ExpeL [22]. We repeat all experiments five times and report both average and best-case performance.

We report core results in Table 2 across high, medium, and low thinking levels. At the high-thinking level, OpenClaw agents using the insight library achieve an average accuracy

Table 3. Percentage of papers guided by insights generated from different models. FoT improves overall guidance rate over the baseline of independently solving tasks without the library ('Isolated Agents') and the baseline that simply stores previous tasks in RAG for retrieval ('RAG').

Setting	Model	Isolated Agents (Gemini)				RAG (Gemini + All Papers)				FoT (Gemini + Insight Library)			
		Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster
1 (23 → 24)	Gemini 3.0 Pro	41.3	**70.9**	**73.8**	33.3	66.5	58.1	69.2	66.4	**67.4**	67.4	68.7	**67.2**
2 (24 → 25)	Gemini 2.0 Flash	0.7	0.5	0.8	0.7	77.4	63.4	67.4	79.6	**82.4**	**82.2**	**82.1**	**82.4**

that exceeds even the best isolated-agent accuracy. The insight library can also lower overall token consumption in the Gemini 3.1 Pro (High) and Gemini 2.5 Flash Lite experiments. Similar to the transferability experiments in Section 4.1, FoT enables medium- and low-thinking agents to reach performance comparable to isolated agents at high and medium thinking levels, respectively. In the Gemini 2.5 experiments, most of the traces come from faulty reasoning, yet FoT is not misled and still achieves accuracy above 50%, demonstrating robustness to noisy traces. Results of comparing with two additional enhanced isolated reasoning baselines and ExpeL [22] are in Appendix D.3.2, where we again observe the significant superiority of FoT, especially when solving cross-domain tasks.

Since these tasks involve multi-round interactions and tool use rather than simple answer generation, we also measure wall-clock execution time, defined as the time from sending a query to the agent until task completion, including tool execution. We observe that latency decreases when the thinking level is increased from low to high, even though higher thinking levels generate more tokens due to more efficient calling of suitable tools. More details and two concrete examples to illustrate how agents with and without the library handle these cases are in Appendix D.3.3.

4.4 Research Insight Discovery

We demonstrate that with FoT, agents collectively extract and synthesize insights across diverse areas of machine learning research. This differs from work on end-to-end agentic workflows that execute the full scientific pipeline and generate submission-ready papers [3, 4, 13, 44–47], which is not what we focus on here. In our setting, each agent is assigned one or more papers to read and identify research questions. FoT then curates an insight library of actionable and generalizable research insights. We evaluate how many future papers could have been guided by these insights as core contributions, using four criteria: concrete methodological usage, how the method is presented, whether the paper's core contribution would fundamentally change without the insight, and whether the insight is novel; see Prompt 6 in the appendix. Note that methods requiring explicit labeling cannot handle such applications, as there is no explicit success signal for reading papers.

We consider two settings depending on whether the base LLM may have seen the evaluation papers during pretraining. In Setting 1, 1574 agents each read one accepted ICLR 2023 paper, and we evaluate on accepted ICLR 2024 papers using Gemini 3.0 Pro[2] for all agents and the server. Since this model may already contain knowledge of ICLR 2024 papers, Setting 2 uses Gemini 2.0 Flash[3] to read ICLR 2024 papers, and we evaluate on ICLR 2025 papers. Gemini 2.0 Flash has no access to 2025 papers because its cutoff date is before the ICLR 2025 submission deadline.

We compare against two baselines. First, we query a standalone Gemini model to generate the top 200 insights using ICLR keywords and the same insight definition, then evaluate whether these insights guide papers in the following year. Second, we use retrieval-augmented generation (RAG) [48] to build a datastore of all selected papers from the previous year and ask the LLM to generate the top 200 insights using retrieved content as reference. These baselines test whether FoT improves insight discovery beyond the LLM's standalone capability and simple retrieval over prior papers. By default, we use Gemini 3.0 Pro as judge and report results in Table 3. We further cross-validate with Gemini 2.5 Flash Lite, DeepSeek-R1-Distill-Qwen-7B, and Qwen2.5-7B-Instruct in Appendix D.4.1. FoT insights achieve the highest guidance rate across evaluations by all four models. For instance, when evaluated by Gemini 2.5 Flash Lite, directly prompting Gemini 3.0 Pro guides only 2.2% of ICLR 2024 papers, whereas FoT insights guide 70.6%. In addition to LLM-based judging, we manually verify the generated insights and provide a subset in Appendix H.1.

A potential concern with centralized agentic systems, such as server-deployed auto-research agents, is that they may require access to full research papers. For unpublished or unlicensed papers, such sharing can raise intellectual property concerns. In contrast, FoT uploads only abstract metacognitive summaries, as illustrated in Appendix H.3, without exposing paper details or sensitive experimental data (Section 5). This design allows users to leverage local agentic assistants while still benefiting from global insights produced

[2]We use the checkpoint gemini-3.0-pro-preview released on November 18, 2025.

[3]We use the checkpoint gemini-2.0-flash-001 released on April 9, 2025.

Table 4. Evaluation of decoupling insight library models and reasoning models on math problem-solving. Insights produced by stronger models (Gemini 3.0 Pro) consistently benefit weaker agents (DeepSeek), while libraries from weaker agents can also improve stronger ones.

Reasoning Model	Insight Library Model	Avg	# Output Tokens	# Loops
Isolated DeepSeek	N/A	0.537	6312	7
DeepSeek	Gemini 3.0 Pro	**0.625**	**4317**	**4**
Isolated Gemini 3.0 Pro	N/A	0.846	891	3
Gemini 3.0 Pro	DeepSeek	**0.902**	**745**	**0**

by the server. The shared insight library further supports iterative improvement, helping both agents and humans refine ideas and develop future research.

5 Properties of FoT

Transferability Across New Models and Tasks.

We investigate whether (1) the insight library curated by one LLM model is compatible with a different model for reasoning, and (2) whether the library curated in one application can be transferred to new, unseen tasks. Table 4 demonstrates that FoT allows for the decoupling of the base LLMs for actual reasoning and the models for building the shared insight library. Moreover, we observe that an insight library generated by a weaker model (DeepSeek) can supervise and improve the performance of stronger ones (Gemini 3.0 Pro) (last two rows of Table 4). This indicates that a variant of 'weak-to-strong generalization' can happen in the text space, beyond the gradient space studied in prior works [e.g., 49].

We further investigate how the insight library generalizes to unseen tasks that are not involved in the library construction process. For mathematical problems and multi-domain collaboration benchmarks, we split the tasks into two disjoint halves: one half is used for insight library construction, while the other half is reserved exclusively for evaluating the library. For real-world daily tasks, we use the old 23 tasks from the evolving PinchBench V1 for insight library construction, and evaluate generalization on the newly introduced 124 tasks spanning 53 categories in PinchBench V2. Same as the previous settings, we conduct mathematical problem solving and multi-domain collaboration experiments using 3 different random seeds, and evaluate real-world daily tasks using 5 different random seeds.

We observe that FoT achieves the largest performance gains during test-time learning, while still maintaining consistent improvements on unseen tasks. This demonstrates that the insight library can continuously benefit newly introduced agents, enabling knowledge transfer and generalization beyond the tasks used during library construction. Complete results on each benchmark are in Appendix E.1. In Appendix E.1, we also verify that new agents on unseen tasks in Claw-Eval [50] (199 agents) can benefit from an insight library generated on PinchBench (23 agents). The library transfers effectively across agents and tasks, improving performance while reducing token usage.

Safeguarding Raw Problem Instances. In addition to improving reasoning effectiveness and efficiency, FoT brings about the side benefit of safeguarding raw problem instances. *First*, we observe that the uploaded abstracted reasoning traces do not contain original problem or task descriptions. We employ a strong multi-turn prompt-stealing attacker [51] to reconstruct the agent input from given reasoning traces. The token-level F1 scores between the reconstructed and original inputs are smaller than 0.25. Also, no personal identifiable information (PII) is leaked. *Second*, we verify that the insight library does not contain answers to the questions, preventing agents from directly retrieving answers via shortcuts. Using shingle intersection and a strict 4-gram matching, we compute Jaccard similarity between the insight library and ground-truth answers, as well as between reasoning traces and input questions. We observe near-**zero** similarity. Full details and quantitative results are provided in Appendix E.2.

6 Ablation Studies

In this section, we empirically study the effects of key FoT parameters. Sections 6.1 to 6.3 examine the impacts of varying local reasoning strategies, server aggregation strategies, and heterogeneous base LLM models, respectively. We further study FoT performance under different library sizes (Section 6.4) and different agent participation scenarios (Section 6.5).

6.1 Effects of Local Reasoning Strategies

In this section, we conduct ablations to (1) examine how our reasoning trace extraction differs from existing mechanisms, and (2) verify that FoT is flexible and supports a variety of local reasoning and self-improvement approaches. We compare with the original prompts (Appendix G.3) in work of Didolkar et al. [23], which we adapt from, and other agent reasoning and self-improvement approaches such as Agentic Context Engineering (ACE) [26], HyperAgents [25], and Evolving Prompt [28]. A subset of core results are reported in Table 6 and detailed implementation and full results on more applications are provided in Appendix F.1.

We see that FoT with different local reasoning methods can improve over their 'Isolated Agents' baseline where each agent solves tasks using the corresponding reasoning method separately. Using our designed prompts improves over variants with other local reasoning methods in most cases, yielding an average +8.1% accuracy gain, along with fewer generated tokens and reasoning loops (Appendix F.1). FoT with HyperAgents as the local reasoning method serves as a strong alternative in some cases. A potential concern with HyperAgents is that uploading evolved prompts may

Table 5. Average accuracies across different tasks and experimental settings on existing and new, unseen tasks. For FoT, agents utilize the shared insight library for both sets of problems, whereas the isolated reasoning agents perform inference independently without access to any shared library.

		Mathematical Problem Solving		Multi-Domain Collaboration	Real-World Daily Tasks	
	Algorithm	Gemini 3.1 Pro	Gemini 2.5 Flash Lite	Gemini 2.5 Flash Lite	Gemini 3.1 Pro	Gemini 2.5 Flash Lite
Existing Problems	Isolated Agents	88.50±1.80	88.24±3.51	82.36±1.88	76.29±3.39	20.12±6.21
	FoT	90.10±0.89	90.12±2.00	85.00±0.60	84.82±2.33	52.11±1.26
New Problems	Isolated Agents	89.13±2.19	87.33±1.64	84.96±1.53	85.03±1.51	34.60±1.42
	FoT	89.66±1.22	89.94±0.81	86.60±0.29	86.67±1.05	36.67±0.82

Table 6. Average accuracies of using different local reasoning strategies in each algorithm in math problem solving tasks (LiveMathBench).

Base Models	Algorithms	Local Reasoning	Avg Acc.
DeepSeek-R1-Distill-Qwen-7B	Isolated Agents	plain inference	0.537
	FoT	Ours (Prompts 2-3)	**0.553**
	Isolated Agents	Metacognitive Reuse [23]	0.464
	FoT		**0.595**
Gemini 3.0 Pro	Isolated Agents	plain inference	0.846
	FoT	Ours (Prompts 2-3)	**0.928**
	Isolated Agents	Metacognitive Reuse [23]	0.850
	FoT		**0.905**
	Isolated Agents	HyperAgents [25]	0.850
	FoT		**0.912**
	Isolated Agents	ACE [26]	0.375
	FoT		**0.502**
	Isolated Agents	Evolving Prompts [28]	0.370
	FoT		**0.451**

introduce privacy risks, as these prompts can contain descriptions of the underlying questions.

6.2 Effects of Global Aggregation Strategies

We demonstrate that FoT is flexible to support different global aggregation algorithms on the server side. In particular, we evaluate the baseline of simply appending the shared traces, and other context compression techniques, including Chain-of-Density (CoD) [33] and Context Compaction from Claude Cookbook [34] (denoted as 'Compaction'). For all three baselines, we vary the corresponding compression ratio parameters to illustrate the tradeoffs between task performance (accuracies or research idea guidance rate) and efficiency (number of reasoning tokens as well as library sizes).

We see that on math problem-solving, simply concatenating all traces results in the worst accuracy. Even when the total tokens remain within the context window, long prompts can still lead to 'context loss', observed by previous works [e.g., 52]. Varying the percentage of traces to append

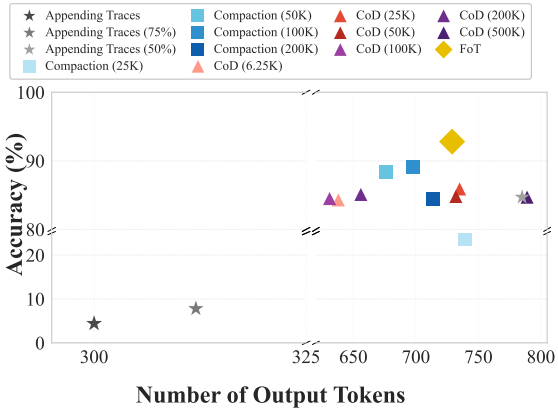

Figure 5. Accuracy and number of output tokens using different server-side algorithms on LiveMathBench. FoT achieves the highest accuracy with a moderate number of tokens, providing a good trade-off between accuracy and efficiency.

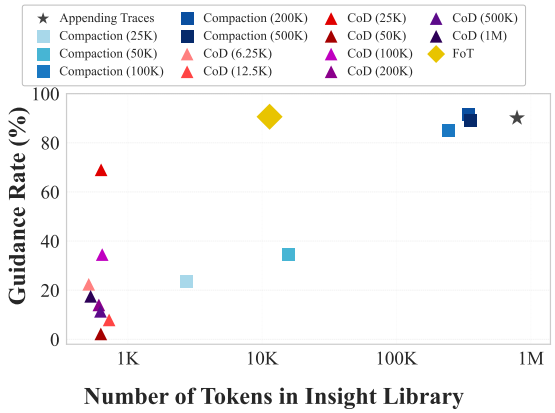

Figure 6. Percentage of papers guided by insights using different server-side aggregation. FoT generates a reasonably-sized library guiding most papers. General context compression methods might not be the most suitable for the task.

(75% and 50% randomly picked) does not help much. CoD and Compaction (plugged into FoT) can be effective and yield performance gains over the independent reasoning baselines, although the accuracy improvement is smaller than that achieved by our proposed prompts. In the ML insight discovery task (Figure 6), we observe that CoD adopts an aggressive context compression strategy and thus produces far fewer tokens. However, the generated insights are not sufficiently satisfactory. For Compaction, the guidance rate increases as the library size increases, but achieving a good guidance rate requires a large number of tokens. FoT with our designed aggregation captures shared patterns without over-generalizing and produces more meaningful insights at a modest size.

6.3 Effects of Heterogeneous Local and Server Agent Models

We further demonstrate the flexibility of FoT in how it is compatible with different agent models deployed on the local clients or the server. Full results are provided in Appendix F.2. On math problem solving, with Gemini 3.0 Pro as local agents and DeepSeek as the server model, FoT reaches 88.1%, compared with 84.6% for isolated reasoning and 92.8% when both server and agents use Gemini 3.0 Pro. With DeepSeek as local agents and Gemini 3.0 Pro as the server, FoT reaches a 58.0% accuracy, compared with 53.7% in isolation and 55.3% when both sides use DeepSeek. These results suggest that FoT remains effective across heterogeneous LLM configurations, with larger gains when the server model is stronger. Similar conclusions hold for research insight discovery. Table 25 in Appendix F.2 further demonstrates that when we have local agents each using different LLMs but working on the same task, FoT still provides performance gains. While deploying all modules with the strongest LLM yields the best performance, users can select lighter alternatives and leverage FoT's flexibility in specific scenarios.

6.4 Effects of Insight Library Size

In Figure 7, we vary the insight library size and observe that, without strict constraints on the total number of insights, FoT naturally reaches a favorable tradeoff between library size and performance at 20 insights, as performance improves only marginally when increasing the library size from 20 to 200. Details are provided in Appendix F.3.

6.5 Effects of Agent Participation

We further analyze the impact of varying the number of participating agents in FoT. For research insight discovery, performance improves as the total number of papers read increases. When the total number of papers is fixed, the number of agents has only a minor impact, with more agents yielding slight additional gains. Since these results align with intuition, we defer the details to Appendix F.4.

We further examine how individual agents contribute to the insight library through two settings: 'Include One', where only one fixed local agent continuously contributes reasoning traces while all agents use the resulting library for inference, and 'Exclude One', where the library is constructed from all agents except one. As shown in Table 7, every agent improves performance when acting as the sole contributor, but none matches the performance achieved when all agents contribute, indicating that FoT benefits from broad agent participation. Table 7 also shows that V202505 Hard is the most influential agent, as it yields the largest improvement over isolated agents when included alone and causes the largest performance drop when excluded. We provide detailed performance of each agent in Appendix F.4. In both settings, agents are evaluated on data that are partially unseen during library construction, yet the insight library still improves performance.

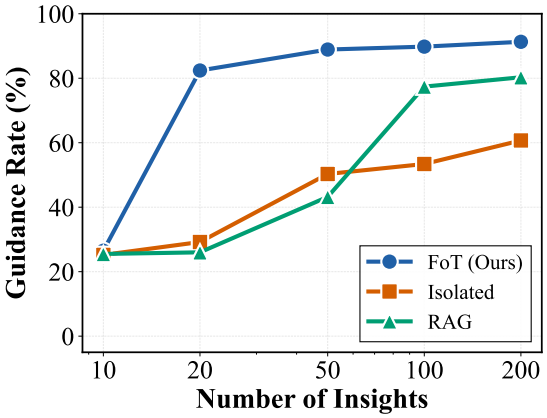

Figure 7. Comparison of FoT, isolated agents, and RAG in their ability to generate insights that can cover the core contributions of ICLR 2025 papers (guidance rate) under varying numbers of generated insights.

Table 7. Changes in average accuracies relative to the Isolated Agents baseline and FoT where all agents participate. Each row corresponds to one agent, identified by the benchmark it works on. **Include One**: The insight library is built from this agent alone. **Exclude One**: The insight library is built from all agents except this one. Changes in average cost is in Appendix F.4.

	Include One		Exclude One	
Agent	Δ Isolated Agents	Δ FoT	Δ Isolated Agents	Δ FoT
AIME24	+1.7	−6.5	+3.9	−4.4
AIME25	+1.9	−6.4	+3.7	−4.5
AMC	+1.5	−6.8	+4.6	−3.7
CCEE	+3.4	−4.8	+3.9	−4.4
CNMO	+0.7	−7.5	+1.5	−6.8
WLPMC	+1.2	−7.0	+3.9	−4.4
V202412 Hard	+1.3	−6.9	+3.3	−5.0
V202505 Hard	+5.2	−3.1	+1.4	−6.8

7 Conclusion, Limitations, and Future Work

In this work, we have proposed the first federated-like multi-agent reasoning framework, Federation over Text (FoT), which enables multiple agents (each solving a task) to collaboratively learn an insight library by sharing and aggregating local reasoning traces iteratively. On the other hand, semantic-level optimization may be a double-edged sword due to inherent LLM hallucinations, which might introduce erroneous reasoning traces into the insight library. Mitigating this fundamental challenge is beyond our current scope, but future work can investigate hallucination propagation within FoT. We have demonstrated the effectiveness of FoT across diverse applications, including mathematical reasoning, cross-domain collaboration, research insight discovery, and daily OpenClaw agent tasks. Further analyses of reasoning traces, extracted insights, and ablation studies reveal how insights guide performance gains and highlight the flexibility of FoT with different local reasoning strategies and server-side aggregation methods. In future work, we aim to automate harness design [53] for FoT. In terms of broader impact, clients may leverage the framework of FoT for good or harmful use cases, such as injecting adversarial reasoning traces into the insights.

As the system scales, more reasoning traces will be uploaded to the server. To manage this growth, we can explicitly constrain the size of the insight library (see Appendix F.3) by continuously merging the existing library with newly generated insights, while allowing the server-side agent to select and retain the most useful insights based on updated reasoning traces. In future work, we will further investigate scenarios where the problem and task distributions encountered by agents drift over time, with a particular focus on how to preserve high-quality and transferable insights while preventing the insight library from being negatively affected by distribution drift across agents.

References

[1] Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. Answering questions by meta-reasoning over multiple chains of thought. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5942–5966, 2023.

[2] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 17889–17904, 2024.

[3] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. arXiv:2408.06292.

[4] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043. Association for Computational Linguistics, 2025.

[5] Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. Researchtown: Simulator of human research community. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 73051–73096, 2025.

[6] Yongjun Zhang. Vibe researching as wolf coming: Can AI agents with skills replace or augment social scientists?, 2026. arXiv:2602.22401.

[7] Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, et al. A survey of frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems, 2025. arXiv:2504.09037.

[8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, 2017.

[9] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:132208–132237, 2024.

[10] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of LLMs, 2025. arXiv:2501.06322.

[11] Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[12] Stephen Chung and Wenyu Du. The station: An open-world environment for AI-driven discovery, 2025. arXiv:2511.06309.

[13] Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, et al. Evoscientist: Towards multi-agent evolving AI scientists for end-to-end scientific discovery, 2026. arXiv:2603.08127.

[14] Zengqing Wu, Run Peng, Shuyuan Zheng, Qianying Liu, Xu Han, Brian I Kwon, Makoto Onizuka, Shaojie Tang, and Chuan Xiao. Shall we team up: Exploring spontaneous cooperation of competing LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5163–5186, 2024.

[15] Qin Zhang, Yangbin Yu, Qiang Fu, Deheng Ye, et al. More agents is all you need. *Transactions on Machine Learning Research (TMLR)*, 2024.

[16] Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[17] Pengsong Zhang, Xiang Hu, Guowei Huang, Yang Qi, Heng Zhang, Xiuxu Li, Jiaxing Song, Jiabin Luo, Yijiang Li, Shuo Yin, et al. aiXiv: A next-generation open access ecosystem for scientific discovery generated by AI scientists, 2025. arXiv:2508.15126.

[18] Chuan Li, Qianyi Zhao, Fengran Mo, and Cen Chen. Fedcot: Communication-efficient federated reasoning enhancement for large language models, 2025. arXiv:2508.10020.

[19] Amirkeivan Mohtashami, Florian Hartmann, Sian Gooding, Lukas Zilka, Matt Sharifi, et al. Social learning: Towards collaborative learning with large language models, 2023. arXiv:2312.11441.

[20] Anthropic. Skills (agent skills repository). http://github.com/anthropics/skills, n.d. GitHub repository.

[21] K-Dense Inc. Claude scientific skills: A comprehensive collection of scientific tools for claude ai. https://github.com/K-Dense-AI/claude-scientific-skills, 2026. skills covering databases, packages, integrations, and analysis tools.

[22] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[23] Aniket Didolkar, Nicolas Ballas, Sanjeev Arora, and Anirudh Goyal. Metacognitive reuse: Turning recurring LLM reasoning into concise behaviors, 2025. arXiv:2509.13237.

[24] Yufan Zhuang, Chandan Singh, Liyuan Liu, Yelong Shen, Dinghuai Zhang, Jingbo Shang, Jianfeng Gao, and Weizhu Chen. Test-time recursive thinking: Self-improvement without external feedback, 2026. arXiv:2602.03094.

[25] Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents, 2026. arXiv:2603.19461.

[26] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models, 2025. arXiv:2510.04618.

[27] Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 27890–27913, 2025.

[28] Jianyu Wang, Zhiqiang Hu, and Lidong Bing. Evolving prompts in-context: An open-ended, self-replicating perspective. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 63036–63087, 2025.

[29] Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[30] Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution, 2026. arXiv:2601.21557.

[31] Zhaofeng Zhong, Wei Yuan, Tong Chen, Xiangyu Zhao, Quoc Viet Hung Nguyen, and Hongzhi Yin. Reasoning pattern alignment merging for adaptive reasoning, 2026. arXiv:2601.03506.

[32] Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. OpenClaw-RL: Train any agent simply by talking, 2026. arXiv:2603.10165.

[33] Griffin Adams, Alex Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. From sparse to dense: Gpt-4 summarization with chain of density prompting. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 68–74, 2023.

[34] Anthropic. Claude cookbooks: Examples and guides for building with claude. https://github.com/anthropics/claude-cookbooks, 2024. Accessed: 2026-03-10.

[35] Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2024, 2024.

[36] Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2025, 2025.

[37] Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are your LLMs capable of stable reasoning? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17594–17632, 2025.

[38] Jiachen Zhao, Yiyou Sun, Weiyan Shi, and Dawn Song. Can aha moments be fake? identifying true and decorative thinking steps in chain-of-thought, 2025. arXiv:2510.24941.

[39] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *Proceedings of the First Conference on Language Modeling (COLM)*, 2024.

[40] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[41] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and et al. Humanity's last exam, 2025. arXiv:2501.14249.

[42] OpenClaw Contributors. Openclaw: Your personal open-source AI assistant. https://github.com/openclaw/openclaw, 2025. Accessed: 2026-04-21.

[43] Kilo.ai. Pinchbench: Benchmarking LLM agents on real-world tasks, 2026. URL https://pinchbench.com/.

[44] Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi, Bodhisattwa Prasad Majumder, Daniel S Weld, and Peter Clark. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13370–13467, 2025.

[45] Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively, 2025. arXiv:2509.26603.

[46] Ruofeng Yang, Yongcan Li, and Shuai Li. Aris: Fully autonomous research via adversarial multi-agent collaboration. https://github.com/wanshuiyin/Auto-claude-code-research-in-sleep, 2026.

[47] Dingjie Song, Hanrong Zhang, Dawei Liu, Yixin Liu, Zongxia Li, Zhengqing Yuan, Siqi Zhang, and Lichao Sun. Dr. claw: An AI research workspace from idea to paper. https://github.com/OpenLAIR/dr-claw, 2026.

[48] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

[49] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[50] Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, and et al. Claw-eval: Toward trustworthy evaluation of autonomous agents, 2026. arXiv:2604.06132.

[51] Zeyang Sha and Yang Zhang. Prompt stealing attacks against large language models, 2024. arXiv:2402.12959.

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

[53] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses, 2026. arXiv:2603.28052.

[54] Eva Y Puspaningrum, Budi Nugroho, Ariyono Setiawan, and Nuraini Hariyanti. Detection of text similarity for indication plagiarism using winnowing algorithm based k-gram and jaccard coefficient. In *Journal of physics: Conference series*, volume 1569, page 022044. IOP Publishing, 2020.

Contents

Abstract 1
A Usage of AI in Writing 12
B Pseudo Algorithms 12
C Hyperparameters 13
D Complete Experiments for Applications of FoT 13
D.1 Mathematical Problem Solving 13
D.2 Multi-Domain Collaboration 13
D.3 Real-World Daily Tasks 14
D.4 Research Insight Discovery 19
E Complete Experiments for Properties of FoT 19
E.1 Transferability 19
E.2 Safeguarding Raw Problem Instances 19
F Complete Experiments for Ablation Studies 21
F.1 Effects of Local Reasoning Strategies 21
F.2 Effects of Heterogeneous Local and Server Agent Models 23
F.3 Effects of Insight Library Size 23
F.4 Effects of Agent Participation 24
G Complete Prompts 27
G.1 Prompts for Our Instantiation of FoT 27
G.2 Complete Problem Prompts 34
G.3 Prompts for Metacognitive Reuse [23] 35
H Insights, Raw Reasoning Processes, and Reasoning Traces 36
H.1 Representative Insights 36
H.2 Complete Raw Reasoning Process for Case Studies in Appendix D.2 39
H.3 Example Reasoning Traces in Section 4.4 43

A Usage of AI in Writing

We employ large language models (LLMs) primarily to improve the grammar and clarity of our writing and to assist with coding. Prompts (Appendix G) used in this work are partially generated by AI with human step-by-step guidance. Due to the nature of this research, insight libraries (Appendix H.1) and reasoning traces (Appendices H.2 and H.3) are AI generated. All research ideas, directions, and decisions, however, are independently conceived and carried out by the authors.

B Pseudo Algorithms

We summarize the pseudo algorithms of federation over (accumulated) gradients and federation over text in Algorithms 1 and 2.

Algorithm 1: Federation over Gradients

```
Input: Initial global model, client datasets
for each iteration do
    for each Client do
        Local optimization starting from the current global model
        Upload new model weights
    end
    Server:
    Collect all local model weights
    Aggregation over model weights
    Broadcast the global model to clients
end
Output: Global model
```

Algorithm 2: Federation over Text

```
Input: Empty insight library, agent tasks, base LLM (can be different across agents)
for each iteration do
    for each Agent do
        Local reasoning given the base LLM and current insight library
        Upload reasoning traces to the server
    end
    Server:
    Gather and aggregate all reasoning traces
    Update the insight library
    Broadcast the new library to agents
end
Output: Insight library
```

C Hyperparameters

In Section 4, we provide detailed descriptions of the tasks, datasets, and federation settings. Here, we additionally report the remaining hyperparameters used in FoT that were not previously introduced. All experimental results are reported with pass@1. In Sections 4.1 and 4.2, experiments are repeated 3 times. In Section 4.3, experiments are repeated 5 times. In Section 4.4, experiments are run once. We set the temperature to 0.7 and top-p to 0.9. We set a repetition penalty of 1.1 for DeepSeek-R1-Distill-Qwen7B and Qwen2.5-7B-Instruct. For all Gemini models, we use the default frequency and presence penalty settings.

For local agents, the maximum output token limits are all set to 32,768 for answer generation, 16,384 for reflection, and 8,192 for reasoning traces. On the server side, the maximum output token limits are set to 32,768 for relationship generation and 16,384 for insight library construction.

All experiments are conducted in FP16 precision using NVIDIA A100 (40GB) GPUs when needed.

D Complete Experiments for Applications of FoT

D.1 Mathematical Problem Solving

Table 8 shows the details of accuracy over each math benchmark. Unless otherwise specified, in all reported experiments, agents solve all benchmark questions in each round and generate corresponding reasoning traces. In other words, agents iteratively solve the exact same benchmark dataset in each round. In experiments from PinchBench to Claw-Eval (Appendix F.2) and "Include One" and "Exclude One" (Appendix F.4), we evaluate performance on tasks where the insight library is built from disjoint tasks. From Table 8, we observe that under both base models, FoT constructs useful insights and improves the average accuracy compared with the baseline of isolated agents solving the task independently. With Gemini 3.0 Pro, the improvement holds for challenging tasks such as AIME25 and V202412 Hard.

D.2 Multi-Domain Collaboration

In the first example, we show that the insight named `Constraint Satisfaction Propagation`, originally derived from AIME25 (a *mathematics* benchmark), is retrieved and applied to a *chemistry* problem in the GPQA Diamond dataset. The key mathematical idea is to model the problem as a constraint satisfaction problem: the goal is to determine whether the `H` atom occupies an axial or equatorial position (i.e., to find a unique valid state, as described in the insight). The agent then tests different isomeric forms and verifies them against thermodynamic constraints.

With the constraints, the agent no longer hallucinates the position of `OH` (in Round 2 of FoT), but correctly identifies the Karplus relationship (in Round 3). The problem instance, our summary of reasoning traces in two rounds, and the insight text that we extract from the library are presented below. We present the complete raw reasoning processes for each round in Appendix H.2.

Question: GPQA Diamond, Problem 89 (Answer: C)

1,3-dibromoadamantane is heated to 240°C with an excess of KOH, creating a light yellow solid product 1, with ^{1}H NMR spectrum: 4.79 (2H), 2.41–2.23 (10H), 1.94 (2H) and a characteristic absorbance in the IR spectrum at 1720 cm^{-1}.

Identify the structure of product 1 and explain the significance of the chemical shifts observed in the NMR spectrum.
This product was then heated with an excess of aluminum isopropoxide, forming product 2. Ozone was bubbled through a solution of 2 at −78°C, followed by addition of dimethyl sulfide, forming product 3. In the ^{1}H NMR spectrum of 3, what is the coupling pattern of the most deshielded hydrogen atom (excluding those that will exchange with deuterated solvent)?
Options:
A. doublet of triplets B. triplet C. triplet of triplets D. pentet

Incorrect (Round 2)

The agent identified the correct synthetic sequence from Grob fragmentation to MPV reduction and then ozonolysis and correctly determined the most deshielded proton was the carbinol H at C7. However, it hallucinated that the OH was in the axial position, leading to a prediction of a "pentet" which are all small couplings.

Correct (Round 3)

While again identifying MPV reduction as thermodynamically controlled yielding the exo-alcohol, it correctly reasoned that in this bicyclic system, the exo-OH means H-C3 is axial. It then applied the Karplus relationship systematically. Axial-axial coupling gives large J (10-12 Hz) while axial-equatorial gives small J (2-5 Hz).

Potentially Related Insights

[Insight] Constraint Satisfaction Propagation: This logic technique solves puzzles by iteratively narrowing the possibilities for each variable based on the restrictions imposed by others. It propagates definite values to trigger a chain reaction of deductions. [When to use] In logic puzzles (Sudoku), scheduling, or system configuration problems where variables must satisfy a web of mutual constraints to find a unique valid state.

[Insight] Kinetic Vs Thermodynamic Control: This chemical principle predicts the major product of a reaction based on conditions (Temperature, Time). Kinetic control favors the fastest-forming product (lowest activation energy), while thermodynamic control favors the most stable product (lowest final energy). [When to use] In organic reactions (like additions to dienes, enolate formation) where multiple products are possible, to determine which isomer dominates at low temps (Kinetic) versus high temps (Thermodynamic).

In the second example, we look into a math problem from AIME. The problem statement, summary of reasoning processes, and potentially-related insights are shown below. In Round 1, with an empty insight library (i.e., isolated reasoning), the agent writes out the formulation but immediately attempts to solve the problem using the greatest common divisor (GCD), leading to an incorrect computation due to the complexity of the expression. Inspired by dynamic programming, specifically the idea of decomposing a problem

Table 8. Math problem-solving (LiveMathBench) accuracies under different settings. With the FoT-generated library, reasoning can reach higher accuracies. We also observe that the stronger LLM yields better results. FoT results that are better than the baselines' are bolded.

Base Models	Algorithms	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	Avg
DeepSeek-R1-Distill-Qwen-7B	Isolated Agent	0.500±0.09	0.400±0.09	0.674±0.07	0.841±0.06	0.722±0.01	0.273±0.15	0.524±0.11	0.360±0.05	0.537
	FoT (Round 3)	0.500±0.09	0.400±0.09	**0.717**±0.07	0.841±0.06	0.722±0.01	**0.364**±0.15	0.524±0.01	0.360±0.05	**0.553**
Gemini 3.0 Pro	Isolated Agent	0.967±0.00	0.933±0.00	0.935±0.00	0.864±0.00	0.889±0.01	0.727±0.07	0.762±0.06	0.690±0.02	0.846
	FoT (Round 2)	**1.000**±0.00	**1.000**±0.00	0.935±0.00	**0.909**±0.07	**1.000**±0.00	**0.909**±0.02	**0.952**±0.08	**0.720**±0.03	**0.928**

into smaller overlapping subproblems, in Round 2, the correct reasoning trace instead breaks the problem down by expressing z_k in terms of z_{k-1}. By applying a recursive formulation, the solution first derives the general term and then substitutes the numerical values, ultimately obtaining the correct answer.

Question: AIME25, Problem 28 (Answer: 248)

Let the sequence of rationals $x_1, x_2, \ldots$ be defined such that $x_1 = \frac{25}{11}$ and

$$x_{k+1} = \frac{1}{3}\left(x_k + \frac{1}{x_k} - 1\right).$$

x_{2025} can be expressed as $\frac{m}{n}$ for relatively prime positive integers m and n. Find the remainder when $m+n$ is divided by 1000.

Incorrect (Round 1)

The model set up the recurrence $x_{k+1} = \frac{x_k^2 - x_k + 1}{3x_k}$ and tracked the numerator/denominator sum S_k, finding $S_{k+1} = S_k^2/3$. The modular arithmetic computation proceeded correctly through CRT (mod 8 and mod 125), but made a computational error: $12^{16} \equiv 84 \pmod{125}$, yielding $3 \times 84 = 252 \equiv 2 \pmod{125}$, and final answer 752.

Correct (Round 2)

The model applied a fundamentally different approach with $z_k = \frac{2x_k - 1}{x_k + 1}$ that transforms the recurrence into the simpler form $z_{k+1} = z_k(z_k - 1)$. The substitution revealed that the denominator follows $D_k = 12^{2^{k-1}}$, and $m + n = 3D_{2025}$. The mod-125 computation now gives $12^{16} \equiv 41 \pmod{125}$, so $3 \times 41 = 123$, yielding final answer 248.

Potentially Related Insight

[Insight] Dynamic Programming State Decomposition: This computational technique solves complex problems by breaking them down into overlapping sub-problems and storing the results (memoization). It relies on defining a 'state' that captures all necessary history. [When to use] In optimization or counting problems involving sequences, grids, or recursive structures where a greedy approach fails (e.g., shortest paths with constraints, knapsack problems, string alignment).

When examining incorrect reasoning processes (e.g., Reasoning Processes 1 and 3 in the Appendix), an interesting phenomenon occurs: there are steps that are totally irrelevant to the final answer. For example, Reasoning Process 3 first gives a complicated formulation of x_{k+1} that does not contribute to the calculation of S_{k+1}, since the formulation of S_{k+1} does not come from the first step. As a result, the first step can be viewed as a 'decorative thinking' step. Instead, with FoT's insight library, agents do not have to exhibit such decorative thinking in Reasoning Process 4. This indicates that insights as a strong and effective prior can help LLM agents quickly start from knowledge of previous problems, reducing the chance of falling into unnecessary or illogical steps. The experiment of measuring true-thinking score [38] further verifies this argument.

For multi-domain collaboration, we supplement our experiments by incorporating multimodal inputs (including images) on challenging tasks where agents cannot solve most questions (i.e., isolated agent accuracy is below 50%). We use the Humanity's Last Exam (HLE) benchmark [41]. We deploy three agents (Gemini-3.0-Pro) to work on HLE, GPQA-Diamond, and V202505-Hard, respectively. Isolated agent accuracy is below 50% for HLE. Table 9 presents the performance gains achieved by FoT on these difficult tasks across different domains. HLE contains 2,500 questions, GPQA-Diamond contains 198 questions, and LiveMathBench Hard 2025 contains 100 questions.

As a result, we observe that when the server aggregates traces, most traces originate from incorrect answers. However, these erroneous experiences do not degrade the quality of the insight library; FoT can still aggregate useful information and improve overall performance.

D.3 Real-World Daily Tasks

D.3.1 Cost Evaluation for the Application of OpenClaw. As agents using API services usually incur monetary costs, cost is a critical concern. In this section, we study whether the additional steps, including reflection and reasoning trace generation for local agents, as well as server-side aggregation of the insight library, introduce significant extra cost. We use the Gemini-3.1-Pro-Preview pricing rules.[4] For isolated agents, the cost depends on the number of input and output tokens. For FoT, we calculate the token usage for reflection and reasoning trace generation, the server-side aggregation, and the input and output tokens when agents use the generated insight library to solve tasks. We also compute

[4] https://ai.google.dev/gemini-api/docs/pricing

Table 9. Performance and computational cost comparison between Isolated Agents and FoT on HLE, GPQA Diamond, and LiveMathBench Hard 2025.

	HLE			GPQA Diamond			LiveMathBench Hard 2025		
Method	Acc. (%)	#Tok.	Loops	Acc. (%)	#Tok.	Loops	Acc. (%)	#Tok.	Loops
Isolated Agents	33.40±3.11	11K±1	0±0	91.90±0.99	769±13	0±0	69.00±0.00	739±7	1±0
FoT	**39.92**±**1.78**	**10K**±1	0±0	**94.42**±0.00	**675**±17	0±0	**71.00**±0.00	739±4	**0**±0

the cost for the setting where agents only use the generated library during inference.

The cost is summarized in Table 10. We observe that although FoT requires more input tokens compared with solving tasks separately (since the prompt includes the insight library), the overall cost is reduced when considering FoT (inference) as fewer output tokens are generated given the generated library. The additional cost introduced by FoT is also not substantial, increasing the total cost only marginally. We believe users would be willing to pay a small additional cost (e.g., less than half a dollar) to achieve higher task accuracies.

Table 10. Cost (USD) for using Gemini-3.1-Pro-Preview under different thinking levels and algorithms. Each cell reports the [min, max] range of cost observed over 5 runs of the experiment for all 25 tasks in PinchBench. FoT (all) calculates all token usage from reflection, reasoning trace generation, and server aggregation to inference with the insight library. FoT (inference) only contains the cost of inference with access to the insight library.

Thinking Level	Isolated Agents	FoT (all)	FoT (inference)
High	[0.62, 0.71]	[1.09, 1.14]	[0.60, 0.64]
Medium	[0.46, 0.63]	[0.86, 1.12]	[0.52, 0.61]
Low	[0.55, 0.62]	[0.95, 1.12]	[0.55, 0.62]

D.3.2 Comparing with Additional Baselines. In addition to isolated agents, we consider two baselines for each local agent. (1) Each OpenClaw agent is equipped with a set of well-defined skills. (2) We collect all raw reasoning histories into a RAG datastore, which is then used by each agent to retrieve past traces when solving tasks. These two strong baselines are designed to highlight the performance advantage of FoT. They also reflect contrasting design philosophies: (1) embedding predefined skills introduces human-curated external knowledge, whereas FoT emphasizes autonomous generation of reusable insights; (2) directly sharing raw reasoning histories conflicts with our objective of not sharing raw questions. For skills, we install all the 52 skills provided by official OpenClaw [5]. Among these skills, skills `nano-pdf`, `summarize`, `github`, `notion`, `slack`, `weather`, and `coding agent` are highly relevant to PinchBench. From Table 11, both skill-augmented agents and those using a RAG datastore improve performance for the Gemini 2.5 model and can also enhance efficiency. However, analysis of the execution logs shows that these baselines rely more heavily on extended reasoning rather than effective tool usage compared to FoT, resulting in higher generation token counts by Gemini 2.5 Flash Lite. In contrast, FoT more reliably invokes the appropriate tools, leading to improved performance.

We also compare FoT with ExpeL. ExpeL similarly lets an agent run some tasks and saves past experiences. However, ExpeL is initially designed for a single agent system and relies on labelling. ExpeL first needs several training tasks and saves the successful trajectories. For failed tasks, the agent will reflect on them with more attempts. Apart from raw successful trajectories, ExpeL also lets the agent reflect on successes and failures to generate several rules. We convert ExpeL into a distributed multi-agent version where we choose OpenClaw agents working on tasks 1-12 on PinchBench as training agents and the rest as evaluation agents. We tune the hyperparameters and set the max reflection depth as 3. The maximum number of rules is 20 since FoT generates 20 insights. The top-K to retrieve from the RAG datastore is 3. For a fair comparison, we first generate an insight library with agents working on tasks 1-12 in FoT and let the remaining agents use the insight library to resolve the tasks.

From the results in Table 12, we find several pieces of evidence showing that ExpeL is not suitable for this application. First, ExpeL is initially focused on single-agent systems and collects experience from the same type of tasks. In PinchBench, tasks are heterogeneous and often unrelated. For example, task 13 involves file searching, task 14 involves generating AI images, and task 15 involves humanizing AI-generated blogs. In contrast, FoT is designed to learn cross-domain knowledge and can achieve performance improvements across different task types. Second, ExpeL relies on labeling and training. This training process incurs substantially higher costs, more than double the total cost of FoT. Third, as discussed earlier, directly extending ExpeL into a distributed version requires uploading raw trajectories.

[5] https://github.com/openclaw/openclaw

Table 11. Performance (pass@1) and efficiency when isolated OpenClaw agents and FoT solve all tasks in PinchBench. FoT significantly improves accuracy while reducing generation token usage. We further compare against two stronger baselines: (1) isolated agents equipped with predefined, well-specified skills, and (2) isolated agents augmented with a RAG datastore constructed from the reasoning histories of all agents.

LLM (Thinking Level)	Algorithms	Score (out of 100)			Execution Time (s)			# Generation Tokens		
		Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
Gemini 3.1 Pro (High)	Isolated Agents	80.50	76.29	72.21	1864.74	**1764.72**	1683.61	53K	50K	46K
	Isolated Agents with Skills	81.93	65.27	54.75	3336.35	2808.19	2062.83	52K	**46K**	42K
	Isolated Agents (RAG)	83.07	64.08	52.76	3244.99	3095.42	2817.16	**51K**	48K	**41K**
	FoT (Round 2)	**87.32**	84.82	81.71	**1845.03**	1787.98	**1658.75**	52K	48K	46K
Gemini 2.5 Flash Lite	Isolated Agents	27.90	20.12	12.70	2932.16	2835.36	**2157.49**	57K	55K	54K
	Isolated Agents with Skills	33.63	32.58	30.07	3329.39	3014.51	2898.64	77K	70K	59K
	Isolated Agents (RAG)	28.52	27.41	25.76	**2980.11**	**2783.53**	2492.54	65K	60K	57K
	FoT (Round 2)	**53.29**	**52.11**	**50.36**	3667.24	3466.19	3223.15	57K	**54K**	**52K**

Table 12. Performance (pass@1) and efficiency when OpenClaw agents and FoT solve tasks 13-23 in PinchBench. ExpeL uses agents working on task 1-12 as training agents while FoT uses them to generate insight library.

LLM (Thinking Level)	Algorithms	Score (out of 100)			# Generation Tokens			Total Cost (USD)		
		Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
Gemini 3.1 Pro (High)	Isolated Agents	72.45	66.32	59.98	33K	29K	26K	**0.38**	**0.35**	**0.33**
	ExpeL	75.00	57.76	31.90	51K	49K	46K	1.67	1.56	1.53
	FoT	**90.68**	**77.05**	**64.50**	**30K**	**28K**	26K	0.77	0.67	0.62

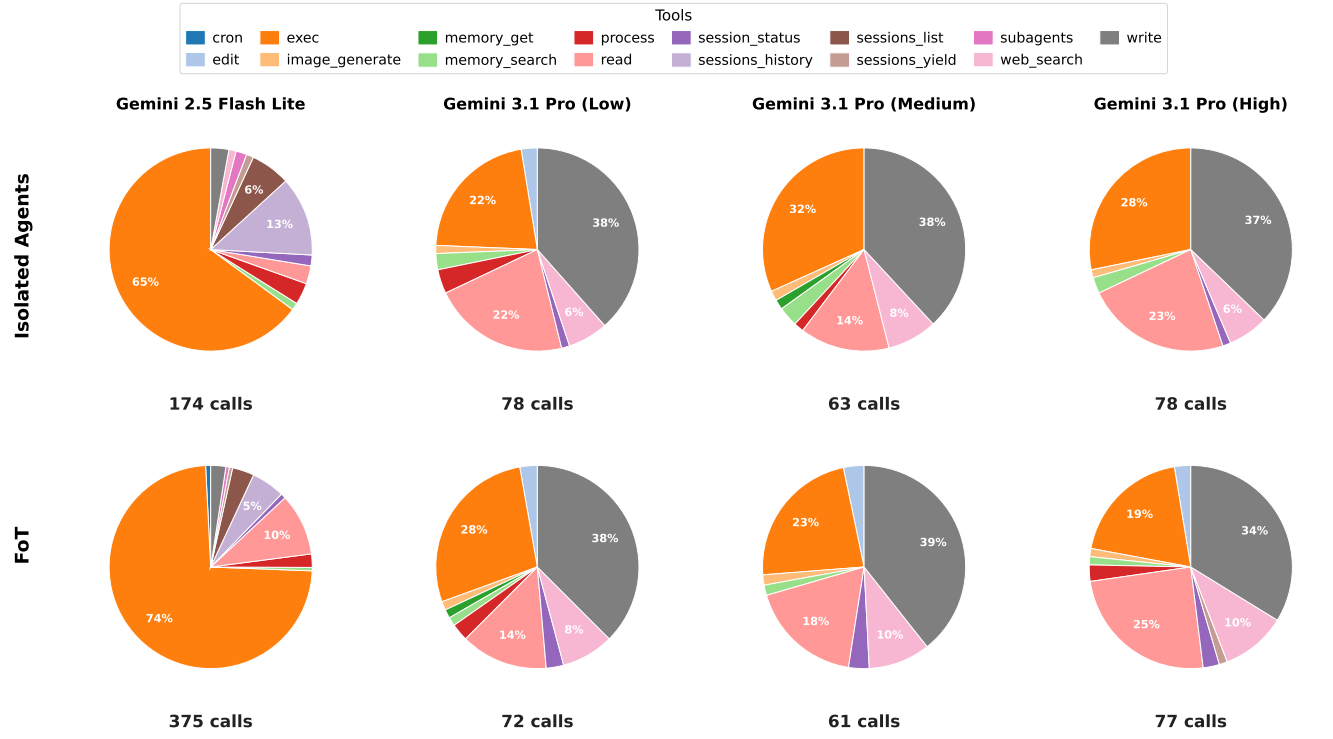

Figure 8. Comparison of tool-call counts across different models and algorithms on the 23 tasks in PinchBench v1. For each setting, we sample the run with the minimum execution latency. In Gemini 3.1 Pro, higher thinking levels tend to reduce execution-time tool calls, and FoT further decreases tool usage under the medium and high thinking settings.

D.3.3 Examples of Tool Usage. As shown in Figure 8, for Gemini 3.1 Pro, the number of execution-tool calls decreases as the thinking level increases, suggesting that the model can more effectively identify the most critical tools and solve the task with fewer redundant calls. This also explains why FoT improves both accuracy and latency under the medium and high thinking settings: agents can avoid repeated or unnecessary tool execution by leveraging the shared library.

The first example is the 'Search and Replace in Files' task, with task ID `task_skill_search`, from PinchBench. We use Gemini 3.1 Pro with the medium thinking level. To more clearly demonstrate the workflow, we show each tool-calling step together with its arguments and outputs, while omitting the model's reasoning outputs. With the hint from the insight library, agents directly use functions of READ and EDIT to search and write the related information rather than using EXEC function with system functions of 'cat'. In this way, the agent can organize the editing in a more structured way and resolve the task more efficiently.

Potentially Related Insight

```
[insight] Constraint Driven SchemaMapping: This organizational
paradigm translates complex, multi-part free-text instructions into
a rigid structural blueprint before generation begins, forcing
complete adherence to constraints. It solves the problem of
generative models silently dropping minor prompt requirements or
hallucinating output shapes. When to use: Apply when generating
highly structured reports, standardized data pipelines, or tasks
that explicitly demand specific formats, exact paragraph counts, or
multifaceted comparative tables. Details: 1) Parse the user prompt
to identify all numerical, structural, and topical constraints. 2)
Map these constraints into an internal document schema (e.g.,
specific markdown headers, predefined table columns). 3) Execute a
thematic bucketing pass over the source data, filling the
pre-established schema. 4) Cull objective constraints (e.g.,
filtering a list of 50 items down to exactly the 5 requested) before
finalizing the write step.
[insight] Idempotent Environment Scaffolding: This technique focuses
on safely injecting configurations or code into environments by
using implicit creation tools and context-aware, non-destructive
mutation strategies. It prevents overwriting existing work or
failing due to missing parent structures. When to use: Apply when
setting up project directories, dynamically generating logs, or
appending multiline configurations (YAML, JSON) across
multi-environment workspaces. Details: 1) Rely on write operations
that inherently construct missing parent directories, halving the
needed operation count. 2) Enforce OS-agnostic pathing (forward
slashes). 3) Check if the target is populated; if so, switch from
'overwrite' to 'append' or use format-aware text replacement that
respects specific syntax (like YAML indentation or JSON quotes)
rather than naive global string replacement. [insight] Verifiable
State Pipelining: This methodology enforces a strictly linear
progression for data mutation: read source, process internally,
write to destination, and physically verify the output state (e.g.,
via byte-count validation or hash checking) before concluding the
task. It solves the problem of automated system hallucination and
unconfirmed executions. When to use: Use whenever performing
destructive file modifications, critical data migrations, or format
conversions where silent failures would compromise system integrity.
Details: 1) Read the existing state into working memory. 2) Apply
processing logic. 3) Execute the write tool. 4) Use the byte-count
or status return of the write operation as an implicit proxy
constraint check. 5) Only report success after the verification step
confirms the structural integrity of the target.
```

Isolated Agents

Call 1. function name: exec.
argument: ls -la config/.
output:

```
drwxr-xr-x 2 user user 4096 Apr 13 20:47 .
drwxr-xr-x 5 user user 4096 Apr 13 20:47 ..
-rw-r--r-- 1 user user  268 Apr 13 20:47 database.yml
-rw-r--r-- 1 user user  304 Apr 13 20:47 settings.json
```

Call 2. function name: exec.
argument: cat config/database.yml && echo "—" && cat config/settings.json.
output: both files printed back-to-back (575 bytes total); the agent now has the YAML and the JSON in working memory but only as flat text.
Call 3. function name: exec.
argument:

```
sed -i 's|http://localhost:3000|https://api.example.com|g'
config/settings.json \
 && sed -i 's/localhost/prod-db.example.com/g'
 config/settings.json \
 && sed -i 's/myapp_dev/myapp_prod/g'
 config/settings.json \
 && sed -i 's/myapp_test/myapp_prod/g'
 config/settings.json \
 && sed -i 's/"level": "debug"/"level": "warn"/g'
 config/settings.json
```

output: The agent gets no confirmation that anything changed; the URL http://localhost:3000 contains localhost as a substring, so the URL rewrite must run before the generic localhost→prod-db.example.com rule, and the first sed has to switch its delimiter from / to | because the URL itself contains / characters.
Call 4. function name: exec.
argument:

```
sed -i 's/localhost/prod-db.example.com/g' config/database.yml \
 && sed -i 's/myapp_dev/myapp_prod/g'       config/database.yml \
 && sed -i 's/myapp_test/myapp_prod/g'      config/database.yml
```

output: sed sweeps both the development: and test: blocks at once, so the agent cannot tell from the result whether one, both, or zero blocks were modified.
Call 5. function name: exec.
argument: cat config/database.yml && echo "—" && cat config/settings.json.
output: both files re-printed after the rewrite, showing host: prod-db.example.com, database: myapp_prod, "endpoint": "https://api.example.com", "level": "warn". This read-back is the agent's only check that the JSON/YAML survived.

FoT (Round 3)

Call 1. function name: exec.
argument: find config -type f.
output:

```
config/settings.json
config/database.yml
```

Call 2. function name: read.
argument: { path: "config/settings.json" }.
Call 3. function name: read.
argument: { path: "config/database.yml" }.
Call 4. function name: edit.
argument:

```
{
  "path": "config/settings.json",
  "edits": [
    { "oldText": "  \"database\": {\n    \"host\": \"localhost\",\n
    \"port\": 5432,\n    \"name\": \"myapp_dev\",",
      "newText": "  \"database\": {\n    \"host\":
      \"prod-db.example.com\",\n
      \"port\": 5432,\n    \"name\": \"myapp_prod\"," },
    { "oldText": "  \"api\": {\n
    \"endpoint\": \"http://localhost:3000\",",
      "newText": "  \"api\": {\n    \"endpoint\":
      \"https://api.example.com\"," },
    { "oldText": "  \"logging\": {\n    \"level\": \"debug\",",
      "newText": "  \"logging\": {\n    \"level\": \"warn\"," }
  ]
}
```

Call 5. function name: edit.
argument:

```
{
  "path": "config/database.yml",
  "edits": [
    { "oldText": "development:\n  adapter: postgresql\n
    host: localhost\n  port: 5432\n  database: myapp_dev",
      "newText": "development:\n  adapter: postgresql\n
    host: prod-db.example.com\n  port: 5432\n  database: myapp_prod" },
    { "oldText": "test:\n  adapter: postgresql\n
    host: localhost\n  port: 5432\n  database: myapp_test",
      "newText": "test:\n  adapter: postgresql\n
    host: prod-db.example.com\n  port: 5432\n  database: myapp_prod" }
  ]
}
```

The second example is the 'Polymarket + News Briefing' task, with task ID `task_polymarket_briefing`, from PinchBench. We use Gemini 3.1 Pro with the high thinking level. For isolated agents, there are redundant API-parameter exploration steps that do not contribute to retrieving the Polymarket information, as well as two undirected web searches. FoT replaces these calls with a more disciplined broad-then-narrow retrieval pattern, achieving a 4% accuracy improvement.

Potentially Related Insight

[insight] Consolidated Intent Based Querying: This data retrieval strategy synthesizes fragmented, conversational user requests into comprehensive, Boolean-expanded master queries. It captures multiple dimensions (entity, temporal markers, semantic equivalents) in a single dense API call. When to use: Apply this when querying search engines, document databases, or external APIs to dramatically minimize round-trip latency, avoid redundant tool usage, and prevent relevant results from being dropped due to colloquial vocabulary mismatches.
[insight] Iterative Temporal Query Anchoring: This active-search technique injects absolute current-date temporal markers (e.g., specific month/year) into otherwise generic search queries, often utilizing a two-pass refinement strategy: a broad search to discover context, followed by a time-anchored search to fetch the latest developments. When to use: Apply this when a prompt imposes strict recency constraints ('latest news', 'upcoming events') to prevent search tools from natively defaulting to historically authoritative but outdated results.,

Isolated Agents

Call 1. function name: exec.
argument:

```
curl -s "https://gamma-api.polymarket.com/events?\
limit=10&active=true&closed=false\
&order=volume24hr&ascending=false" | jq -r '.[0:3] | .[] | .title'
```

output:

```
US x Iran ceasefire by...?
The Masters - Winner
2026 FIFA World Cup Winner
```

Call 2. function name: exec.
argument: the same curl with limit=5, jq projecting .title + " - Volume: " + (.volume24hr|tostring).
output: the same three titles plus two more, now annotated with 24-hour volume. The first request is effectively re-issued just to add a column.
Call 3. function name: exec.
argument: the same curl with limit=3, jq projecting .title, .description of the top event.
output: the 1.8 KB description of the US-Iran ceasefire market. The agent now has descriptions, but only for the first market and only by re-issuing the same API call a third time.
Call 4. function name: exec.
argument: date "+%A, %B %d, %Y".
output: Friday, April 10, 2026. The agent uses this for the file header but does not re-use the date as a temporal anchor in any of its upcoming searches.
Call 5. function name: session_status.
argument: {}.
output: a context-window diagnostic (Tokens: 14k in / 1.3k out | Context: 20k/200k); does not contribute to the artifact.
Call 6. function name: web_search.
argument: { query: "US Iran ceasefire news" }.
output: "A two-week conditional ceasefire between the United States and Iran was announced on Tuesday, April 7, 2026 . . . ". No count is specified and no temporal anchor is supplied; the agent accepts whatever ranks first.
Call 7. function name: web_search.
argument: { query: "The Masters golf 2026 news" }.
output: tournament background and field information for the 2026 Masters; again no temporal narrowing.
Call 8. function name: web_search.
argument: { query: "2026 FIFA World Cup news" }.
output: a generic overview of the 48-team format. Ambient context, not a "last 48 hours" driver.
Call 9. function name: write.
argument: a 1.2 KB polymarket_briefing.md synthesising the three markets from the outputs of Calls 6/7/8.
output: Successfully wrote 1226 bytes to polymarket_briefing.md. The briefing is graded 66.7%.

FoT (Round 3)

Call 1. function name: exec.
argument:

```
curl -s "https://gamma-api.polymarket.com/events?limit=10
&active=true&closed=false" \
 | jq '[.[] | {title: .title, volume: .volume}] |
 sort_by(.volume) | reverse | .[0:3]'
```

output:

```
[
  { "title": "MicroStrategy sells any Bitcoin by ___ ?",
  "volume": 22098024.99 },
  { "title": "Starmer out by...?",
  "volume": 10608225.14 },
  { "title": "Ukraine recognizes Russian sovereignty \
  over its territory by...?",
  "volume": 2350850.67 }
]
```

Call 2. function name: web_search (broad-driver pass).
argument: { query: "\"MicroStrategy\" sells Bitcoin", count: 3 }.
output: MicroStrategy primarily holds Bitcoin long-term; as of April 6, 2026 the company holds ~766,970 BTC. Surfaces the entity but not the catalyst.
Call 3. function name: web_search (broad-driver pass).
argument: { query: "\"Keir Starmer\" step down OR out OR resign", count: 3 }.
output: Starmer is under pressure due to the Peter Mandelson appointment and the May 2026 local elections. The first pass discovers the catalyst. Note the Boolean expansion of the colloquial market title "Starmer out" into formal vocabulary.
Call 4. function name: web_search (broad-driver pass).
argument: { query: "Ukraine recognizes Russian sovereignty territory", count: 3 }.
output: Ukraine's official non-recognition position; establishes the backdrop but is not yet news-fresh.
Call 5. function name: session_status.
argument: {}.
output: Tokens: 9.7k in / 948 out | Context: 22k/200k.
Call 6. function name: web_search (narrow + temporal anchor pass).
argument: { query: "MicroStrategy Bitcoin news April 2026", count: 5 }.
output: "MicroStrategy resumed its aggressive Bitcoin acquisition in April 2026, purchasing an additional 4,871 BTC between April 1 and April 5, 2026, ~$329.9M at ~$67,718 / BTC." This is the last-48-hours driver that the Isolated trace never found.
Call 7. function name: web_search (narrow + temporal anchor pass).
argument: { query: "Keir Starmer resign Peter Mandelson April 2026", count: 5 }.
output: combines the entity, the catalyst the broad pass surfaced (Mandelson), and the temporal anchor; returns the specific April-2026 political timeline including the May local-elections trigger.
Call 8. function name: web_search (narrow + temporal anchor pass).
argument: { query: "Ukraine Russia peace plan Trump April 2026", count: 5 }.
output: details of the "Trump 28-point peace plan" and ongoing diplomatic efforts in early 2026 — once again, the right driver surfaces only after the second pass.
Call 9. function name: write.
argument: a 2.0KB polymarket_briefing.md that, for each market, juxtaposes "the established fact" against "the unresolved variable" (Mandelson dismissal + May local elections; Saylor's April BTC buys vs. $12.4B paper loss; Trump 28-point plan vs. Ukraine's non-recognition stance).
output: Successfully wrote 2006 bytes to polymarket_briefing.md.
Accuracy: 70.7%.

D.4 Research Insight Discovery

D.4.1 Cross Model Validation for Research Insight Discovery. To validate the reliability of the LLM-as-a-judge framework for research insight discovery, we perform cross-model evaluation using three additional models beyond Gemini 3.0 Pro. As shown in Table 13, Gemini 3.0 Pro exhibits self-evaluation bias, with the guidance rate for isolated agents on oral papers dropping from 70.9% to 5.8% when evaluated by Gemini 2.5 Flash Lite. In contrast, FoT maintains strong guidance rates under cross-model evaluation by Gemini 2.5 Flash Lite, indicating better generalization and reduced bias.

E Complete Experiments for Properties of FoT

E.1 Transferability

In addition to transferring the insight library across agents using different LLMs, we evaluate insight transfer across tasks. Extending the setup in Section 4.3, we introduce new agents operating on Claw-Eval [50]. Claw-Eval contains 199 daily tasks, including tasks of visual input and artifact generation, simulated user interaction, trajectory-level grading, embedded constraint checking, and controlled error injection which PinchBench does not include. The Claw-Eval metrics also measure other dimensions such as robustness and safety besides completion. Upon joining the federation, these agents are initialized with an `insight.md` generated from agents previously working on PinchBench. Results in Table 14 show that new agents need fewer inference tokens and hence save cost with the help of the previous library. We provide detailed results of the transferability experiments on each math benchmark in Table 15.

Table 16 shows the details of accuracy over each math benchmark and Table 17 shows the details over benchmarks in multi-domain collaboration.

E.2 Safeguarding Raw Problem Instances

First, we investigate whether the original task description can be inferred from uploaded reasoning traces. We adopt a strong prompt-stealing attack setting [51], where the attacker or an honest-but-curious server aims to reconstruct the original task prompt from observed traces. In PinchBench daily tasks, the attacker is given a reasoning trace and must infer the agent's input, including the user query and auxiliary information (e.g., documents to be summarized). The attack consists of two components: a parameter extractor, which classifies the prompt type into direct user request, role-based (persona or instructional framing), or in-context (demonstrations or substantial context), and a prompt extractor, which reconstructs the prompt. The two components interact for three rounds to refine the output. We use Gemini-2.5-Pro as the attacking model and evaluate attacks on libraries generated by FoT using Gemini-3.1-Pro (high thinking) and Gemini-2.5-Flash-Lite.

We use several metrics to measure whether the attacker reconstructs the original input. We first evaluate reconstruction quality using character-level similarity, defined as $\frac{2M}{T}$,

Table 13. Percentage of papers guided by insights generated by different models, with cross-validation performed by a separate model acting as an LLM-as-a-judge.

Generation Model	Judge Model	Isolated Agents (Gemini)				RAG (Gemini + All Papers)				FoT (Gemini + Insight Library)			
		Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster	Overall	Oral	Spotlight	Poster
Gemini 3.0 Pro	Gemini 3.0 Pro	41.3	**70.9**	**73.8**	33.3	66.5	58.1	69.2	66.4	**67.4**	67.4	68.7	**67.2**
	Gemini 2.5 Flash Lite	2.2	5.8	2.2	2.8	69.4	72.1	73.8	68.4	**70.6**	**73.3**	**74.1**	**69.7**
	Qwen2.5-7B-Instruct	3.2	2.3	3.5	3.2	0.5	0.0	0.5	0.6	**4.0**	**7.0**	**3.3**	**4.0**
	DeepSeek-R1-Distill-Qwen-7B	17.2	20.2	16.4	17.1	34.4	34.5	34.4	34.4	**43.6**	**46.4**	**42.9**	**43.4**
Gemini 2.0 Flash	Gemini 3.0 Pro	0.7	0.5	0.8	0.7	77.4	63.4	67.4	79.6	**82.4**	**82.2**	**82.1**	**82.4**
	Gemini 2.5 Flash Lite	47.3	53.6	54.2	46.1	**62.2**	34.3	**80.5**	50.0	60.9	**64.8**	58.4	**61.0**
	Qwen2.5-7B-Instruct	0.9	0.0	0.5	1.0	1.3	0.0	1.1	1.4	**5.1**	**5.6**	**5.3**	**5.0**
	DeepSeek-R1-Distill-Qwen-7B	22.0	**22.5**	21.4	22.0	17.4	17.8	16.8	17.4	**22.8**	22.1	**23.7**	**22.8**

Table 14. Evaluation of the benefits of insight libraries generated by PinchBench on Claw-Eval. Insights produced by PinchBench can help new agents on Claw-Eval improve performance, save generation tokens, and cost. The reported generation token usage and cost savings are summation over all tasks. Remaining metrics are averaged over tasks; the standard deviation is computed across five runs.

Agent Model	**Insight Library**	Score	Completion Score	Robustness	Safety	# Generation Tokens	Latency (s)	Cost Saved (USD)
Gemini2.5	–	43.22±0.5	30.20±1.5	**94.7±0.7**	**98.82±1.0**	373K±12	36.99±2.3	0
Flash Lite	PinchBench	**43.73±0.2**	**31.01±1.0**	94.4±0.5	98.2 ± 1.0	**352K±10**	**35.64±2.1**	0.051

Table 15. Evaluation of the benefits of insight libraries generated by different LLMs on math problem-solving datasets. Insights produced by stronger models (Gemini 3.0 Pro) consistently benefit agents with weaker LLMs (DeepSeek), while insight libraries curated from weaker agents can lead to performance improvements for stronger ones.

Agent Model	**Insight Library Model**	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard
Isolated DeepSeek	N/A	0.500	0.400	0.674	0.841	0.722	0.273	0.524	0.360
DeepSeek	Gemini 3.0 Pro	**0.833**	**0.567**	**0.696**	0.841	0.722	**0.455**	0.524	0.360
Isolated Gemini 3.0 Pro	N/A	0.967	0.933	0.935	0.864	0.889	0.727	0.762	0.690
Gemini 3.0 Pro	DeepSeek	**1.000**	0.933	0.935	0.886	**0.944**	0.909	**0.905**	**0.700**

Table 16. Math problem-solving accuracies under different settings and unseen questions. With the FoT-generated library, reasoning can reach higher accuracies. Each cell reports mean±std over 3 runs.

	Base Models	**Algorithms**	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	**Avg**
Existing Problems	Gemini 3.1 Pro	Isolated Agent	$1.000_{\pm 0.000}$	$0.978_{\pm 0.031}$	$0.895_{\pm 0.022}$	$0.894_{\pm 0.021}$	$0.926_{\pm 0.052}$	$0.811_{\pm 0.016}$	$0.903_{\pm 0.004}$	$0.673_{\pm 0.034}$	$0.885_{\pm 0.018}$
		FoT	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.898_{\pm 0.021}$	$0.879_{\pm 0.043}$	$0.963_{\pm 0.052}$	$0.878_{\pm 0.087}$	$0.903_{\pm 0.082}$	$0.687_{\pm 0.034}$	$0.901_{\pm 0.009}$
	Gemini 2.5 Flash Lite	Isolated Agent	$0.978_{\pm 0.031}$	$0.978_{\pm 0.031}$	$0.913_{\pm 0.036}$	$0.864_{\pm 0.064}$	$1.000_{\pm 0.000}$	$0.800_{\pm 0.163}$	$0.833_{\pm 0.047}$	$0.693_{\pm 0.034}$	$0.882_{\pm 0.035}$
		FoT	$1.000_{\pm 0.000}$	$0.978_{\pm 0.031}$	$0.927_{\pm 0.041}$	$0.877_{\pm 0.046}$	$1.000_{\pm 0.000}$	$0.811_{\pm 0.016}$	$0.903_{\pm 0.082}$	$0.713_{\pm 0.047}$	$0.901_{\pm 0.020}$
New Problems	Gemini 3.1 Pro	Isolated Agent	$1.000_{\pm 0.000}$	$0.956_{\pm 0.031}$	$0.942_{\pm 0.041}$	$0.894_{\pm 0.057}$	$0.963_{\pm 0.052}$	$0.833_{\pm 0.000}$	$0.879_{\pm 0.043}$	$0.707_{\pm 0.019}$	$0.891_{\pm 0.022}$
		FoT	$1.000_{\pm 0.000}$	$0.978_{\pm 0.031}$	$0.942_{\pm 0.041}$	$0.909_{\pm 0.037}$	$0.926_{\pm 0.052}$	$0.833_{\pm 0.000}$	$0.879_{\pm 0.043}$	$0.727_{\pm 0.009}$	$0.899_{\pm 0.012}$
	Gemini 2.5 Flash Lite	Isolated Agent	$0.978_{\pm 0.031}$	$0.956_{\pm 0.031}$	$0.942_{\pm 0.041}$	$0.909_{\pm 0.037}$	$0.889_{\pm 0.000}$	$0.778_{\pm 0.079}$	$0.848_{\pm 0.043}$	$0.687_{\pm 0.034}$	$0.873_{\pm 0.016}$
		FoT	$1.000_{\pm 0.000}$	$0.956_{\pm 0.031}$	$0.942_{\pm 0.041}$	$0.909_{\pm 0.037}$	$0.926_{\pm 0.052}$	$0.833_{\pm 0.000}$	$0.909_{\pm 0.074}$	$0.720_{\pm 0.033}$	$0.899_{\pm 0.008}$

where M is the number of matching characters (in contiguous substrings appearing in both strings in the same order) and T is the total number of characters across both strings. We also report token-level F1 score and the extent of personally identifiable information (PII) leakage. Due to the small size of PinchBench (25 tasks), PII is manually annotated. In addition, we report the classification accuracy of the parameter extractor component in the model. The parameter extractor performs worse than random guessing, indicating that it fails to reliably identify the prompt structure. In Table 18,

Table 17. Multi-domain collaboration accuracies under different settings and unseen questions. With the FoT-generated library, reasoning can reach higher accuracies.

	Algorithms	AIME24	AIME25	GPQA	GPQA Diamond	LiveCodeBench	**Avg**
Existing Problems	Isolated Agent	0.978 ± 0.031	0.978 ± 0.031	0.929 ± 0.026	0.888 ± 0.017	0.345 ± 0.050	0.824 ± 0.019
	FoT	1.000 ± 0.000	0.978 ± 0.031	0.941 ± 0.015	0.885 ± 0.021	0.446 ± 0.054	0.850 ± 0.006
New Problems	Isolated Agent	0.978 ± 0.031	0.956 ± 0.031	0.946 ± 0.015	0.906 ± 0.018	0.462 ± 0.056	0.850 ± 0.015
	FoT	1.000 ± 0.000	0.956 ± 0.031	0.953 ± 0.006	0.909 ± 0.018	0.515 ± 0.018	0.867 ± 0.003

we see that both character-level similarity and token-level F1 scores are low, further confirming that the reconstructed prompts do not closely match the originals. Prior work [51] suggests that successful prompt reconstruction typically requires similarity exceeding 60%, which is not observed in our setting.

At the same time, these metrics are not expected to be zero. Reasoning traces necessarily retain some task-relevant information to enable knowledge transfer across agents. If similarities were zero, it would imply that the traces are entirely unrelated to the original tasks, i.e., effectively hallucinated and non-informative. Hence, we can draw the conclusion that reasoning traces are effective at hiding raw problem instances from the server. More importantly, no personal identity information can be inferred.

Second, we adopt a strict n-gram matching test to check: (1) that uploaded reasoning traces do not overlap with the original inputs (e.g., benchmark questions and ground-truth answers), and (2) that the insight library does not contain original inputs.

For each application, we append all problem instances together into a document, and partition the document into 4-gram token shingles. Problems refer to questions, tasks, or papers depending on the application. We do the same for answers, insight libraries, and reasoning traces, separately. Each document is a set of all 4-gram token shingles. We then evaluate similarities between (a) reasoning traces and problems, and between (b) the insight library and problems + answers, by reporting the Jaccard similarities between the two sets of 4-gram tokens in the corresponding two documents. Prior work on shingle-based plagiarism detection shows that the choice of n in n-gram significantly affects similarity estimates, with smaller n leading to inflated overlap due to common phrases and larger n being overly sensitive to minor edits [54]. We adopt 4-gram shingles as a balanced choice that reduces false positives while remaining sensitive to meaningful phrase-level reuse.

In Table 19, we observe that Jaccard similarity is near zero, indicating that the reasoning traces have little overlap with raw user questions and answers, achieving one of the goals of FoT. The zero similarity between the insight library and the problems further suggests that an adversary cannot obtain raw problem instances by examining the library.

Furthermore, our work does not focus on studying prompt leakage or defenses for multi-agent reasoning. Instead, in addition to improving reasoning effectiveness and efficiency, FoT brings about the side benefit of data minimization by exposing the underlying logic-level summarization of the reasoning processes as opposed to raw reasoning data or task data. A more thorough investigation of safety and privacy in FoT is beyond the scope of this work.

F Complete Experiments for Ablation Studies

F.1 Effects of Local Reasoning Strategies

From the results in Table 20, we see that FoT consistently improves reasoning accuracy and efficiency under different local reasoning methods, compared to isolated agents using corresponding local reasoning methods. Complete per-benchmark accuracies are reported in Table 21. We tune the hyperparameters of our baselines and report the best average accuracies we obtain. Lower performance in some baselines does not imply that these methods are generally ineffective; rather, it suggests that their mechanisms are less suited to this setting of insight-library construction. For Agentic Context Engineering (ACE) [26], we observe a known limitation identified in prior work [26]: if the Reflector fails to extract meaningful insights from generated traces or outcomes, the constructed context may become noisy or even harmful. We set the maximum number of reflector rounds to 5. On mathematical datasets, ACE generates some misleading traces, leading to performance drops. FoT, in contrast, can help mitigate such effects. Evolving Prompts [28] has known drawbacks in terms of sensitivity to the in-context prompt template and stability. Despite these limitations, plugging them into FoT still improves performance.

The results in Table 22 show that both our native FoT method and HyperAgents produce the most helpful libraries. The strong performance of HyperAgents suggests that local reasoning methods should be adaptable across domains, especially since FoT involves agents solving diverse tasks. This finding also motivates our future plan to enable each FoT agent to autonomously optimize its harness [53].

Table 18. Prompt stealing attack results. The attacker uses Gemini-2.5-Pro with PinchBench reasoning traces in FoT to reconstruct the original input. The low character-level similarity, low token-level F1, and low parameter-extraction accuracy indicate that the attack fails to recover the original input, suggesting an effective defense. No personal identity information (PII) is leaked. The closest reconstruction achieved by the attacker is inferring that the instruction involves common file names such as *"ai blog.txt"* and *"humanized blog.txt"*.

FoT Model	Char Similarity	Token F1	PII leaked	Parameter Extractor Acc
Gemini 3.1 Pro	14.89%	22.85%	0	26.09%
Gemini 2.5 Flash Lite	11.94%	24.44%	0	21.74%

Table 19. The measurement of overlap between the insight library (IL), reasoning traces (RT), and problems. # Problems refer to questions, tasks, or papers depending on the application. Jaccard similarity is computed between 4-gram token sets constructed from problem statements, answers, reasoning traces, and insight libraries.

Application	Agent Model	#Problems	# Tokens RT	# Tokens IL	# Insights	Jaccard (RT and problems)	Jaccard (IL and problems & answers)
Math Benchmark	Gemini 3.0 Pro	300	53K	12K	37	0.000	0.000
Math Benchmark	DeepSeek	300	33K	3K	11	0.000	0.000
Science+Math+Coding	Gemini 3.0 Pro	881	725K	2.9K	34	0.000	0.000
HLE+Science+Math	Gemini 3.0 Pro	2798	450K	3.2K	40	0.000	0.000
Insights from ICLR 2023	Gemini 3.0 Pro	1574	391K	886	2	0.000	0.000
Insights from ICLR 2024	Gemini 2.0 Flash	2260	441K	1.3K	20	0.000	0.000
PinchBench	Gemini 3.1 Pro	23	11K	3.1K	20	0.006	0.000

Table 20. Average accuracies over math problem solving tasks (LiveMathBench) and efficiency comparison under different settings. With the FoT-generated library, agents can achieve higher accuracies and efficiency relative to solving the problems in an isolated manner. Duplicated loops in reasoning refer to how many times the LLM generates repeated sentences in the reasoning process.

Base Models	**Algorithms**	**Local Reasoning Methods**	**Average Accuracy**	**# Generation Tokens**	**#Duplicated loops in reasoning**
DeepSeek-R1 -Distill-Qwen-7B	Isolated Agents	Plain inference	0.537	6312	7
	FoT	Ours (Prompts 2-3)	**0.553**	**5886**	**6**
	Isolated Agents	Metacognitive [23]	0.464	6543	4
	FoT		**0.595**	**5781**	**1**
Gemini 3.0 Pro	Isolated Agents	Plain inference	0.846	891	3
	FoT	Ours (Prompts 2-3)	**0.928**	**728**	**0**
	Isolated Agents	Metacognitive [23]	0.850	938	1
	FoT		**0.905**	**718**	**0**
	Isolated Agents	HyperAgents [25]	0.850	716	0
	FoT		**0.912**	**682**	0
	Isolated Agents	ACE [26]	0.375	860	4
	FoT		**0.502**	**814**	**3**
	Isolated Agents	Evolving Prompts [28]	0.370	1014	7
	FoT		**0.451**	**941**	**4**

Table 21. Accuracies of math problem solving (LiveMathBench) under different settings. With the FoT-generated library, agents can achieve higher accuracies relative to solving the problems in an isolated manner.

Base Models	Algorithms	Local Reasoning Methods	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	Avg
DeepSeek-R1 -Distill-Qwen-7B	Isolated Agents	Plain inference	0.500	0.400	0.674	0.841	0.722	0.273	0.524	0.360	0.537
	FoT	Ours (Prompts 2-3)	0.500	0.400	**0.717**	0.841	0.722	**0.364**	0.524	0.360	**0.553**
	Isolated Agents	Metacognitive[23]	0.367	0.233	0.696	0.682	0.833	0.364	0.333	0.200	0.464
	FoT		**0.500**	**0.400**	0.696	**0.841**	0.833	**0.545**	**0.571**	**0.370**	**0.595**
Gemini 3.0 Pro	Isolated Agents	Plain inference	0.967	0.933	0.935	0.864	0.889	0.727	0.762	0.690	0.846
	FoT	Ours (Prompts 2-3)	**1.000**	**1.000**	0.935	**0.909**	**1.000**	**0.909**	**0.952**	**0.720**	**0.928**
	Isolated Agents	Metacognitive[23]	1.000	0.967	0.913	0.818	0.722	0.818	0.905	0.660	0.850
	FoT		1.000	0.967	**0.935**	**0.886**	**0.889**	**0.909**	**0.952**	**0.700**	**0.905**
	Isolated Agents	HyperAgents[25]	1.000	1.000	1.000	0.923	1.000	0.456	0.667	0.750	0.850
	FoT		1.000	1.000	1.000	**1.000**	1.000	**0.545**	**1.000**	0.750	**0.912**
	Isolated Agents	ACE[26]	0.600	0.389	0.563	0.591	0.250	0.167	0.182	0.256	0.375
	FoT		**0.714**	**0.500**	**0.700**	0.591	**0.556**	**0.375**	**0.250**	**0.326**	**0.502**
	Isolated Agents	Evolving Prompts[28]	0.233	0.167	0.630	0.659	0.389	0.546	0.143	0.192	0.370
	FoT		**0.300**	**0.233**	**0.674**	**0.796**	**0.556**	**0.636**	**0.191**	**0.220**	**0.451**

Table 22. Paper guidance rate in setting 2 where base models are Gemini-2.0-Flash under different local reasoning methods. Agents read ICLR 24 paper and check whether ICLR 25 papers can be guided by the insight library.

Algorithms	Local Reasoning Methods	Poster	Spotlight	Oral	All
Isolated Agents	Plain inference	0.7	0.5	0.8	0.7
FoT	Ours (Prompts 2-3)	82.4	82.1	82.2	82.4
	Metacognitive Reuse[23]	60.6	56.6	57.7	60.0
	HyperAgents[25]	82.7	80.5	81.2	82.4
	Agentic Context Engineering[26]	5.43	6.05	7.98	5.64
	Evolving Prompts[28]	14.1	12.6	12.7	13.8

F.2 Effects of Heterogeneous Local and Server Agent Models

We report in Table 23 the performance when the local agents are DeepSeek (or Gemini 3.0 Pro) and the server model is Gemini 3.0 Pro (or DeepSeek), respectively. Although using DeepSeek as the server for Gemini 3.0 Pro local agents performs worse than using Gemini 3.0 Pro as the server, it still outperforms the single-agent baseline.

For the task of research insight discovery, we replace the server agent from Gemini 2.0 Flash with Gemini 3.0 Pro. The results Table 24 confirm our conclusion that a stronger server LLM further improves performance.

We also consider the setting where multiple agents solve the same task using different models. We have 4 agents, each agent picks a different model, and they all solve the same LiveMathBench Hard 2025 benchmark. Although FoT is primarily designed to aggregate knowledge across diverse tasks, it still provides benefits in this scenario. As shown in Table 25, FoT improves performance even when agents operate on identical tasks, reflecting one potential realistic setting where multiple agents collaborate on the same problem. Since leveraging multiple agents to collaboratively solve a single task is not our focus, we do not conduct further comparisons with such baselines.

F.3 Effects of Insight Library Size

Here, we demonstrate the effects of insight library size on the ML research insight discovery application. Recall that one baseline is isolated reasoning, and the other baseline is to query the ML research insights through a RAG system, where we store old papers in an external database as reference. In Prompt 3, we replace the phrase 'proper number' with explicit quantities (e.g., 10, 50, 100). From Figure 9, we can see that as the number of insights increases, the guidance rate of all three approaches increases, and FoT outperforms the other two. We note that when FoT does not have strict constraints on the total number of insights, it will automatically arrive at a good tradeoff between the number of insights and performance (20), since the performance gain increases slowly from 20 insights to 200. We can see that RAG can be effective in encoding external information, but it requires maintenance of a potentially huge datastore (storing all the

Table 23. Mathematical problem-solving performance using FoT-generated insight libraries under heterogeneous client and server LLM configurations, compared to isolated agents, with improvements highlighted in bold.

Algorithms	Agent LLM	Server LLM	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	Avg
Isolated Agents	DeepSeek	–	0.500	0.400	0.674	0.841	0.722	0.273	0.524	0.360	0.537
FoT	DeepSeek	Gemini 3.0 Pro	**0.700**	0.400	**0.717**	0.841	0.722	**0.364**	0.524	**0.370**	**0.580**
Isolated Agents	Gemini 3.0 Pro	–	0.967	0.933	0.935	0.864	0.889	0.727	0.762	0.690	0.846
FoT	Gemini 3.0 Pro	DeepSeek	0.967	0.933	0.935	**0.886**	0.889	**0.909**	**0.857**	0.670	**0.881**

Table 24. Paper guidance rate using FoT-generated insight libraries under heterogeneous client and server LLM configurations.

Algorithms	Agent LLM	Server LLM	Poster	Spotlight	Oral	All
Isolated Agents	Gemini 2.0 Flash	–	60.7	60.5	60.6	60.7
FoT	Gemini 2.0 Flash	Gemini 2.0 Flash	82.4	82.1	82.2	82.4
FoT	Gemini 2.0 Flash	Gemini 3.0 Pro	**86.7**	**82.6**	**86.9**	**86.3**

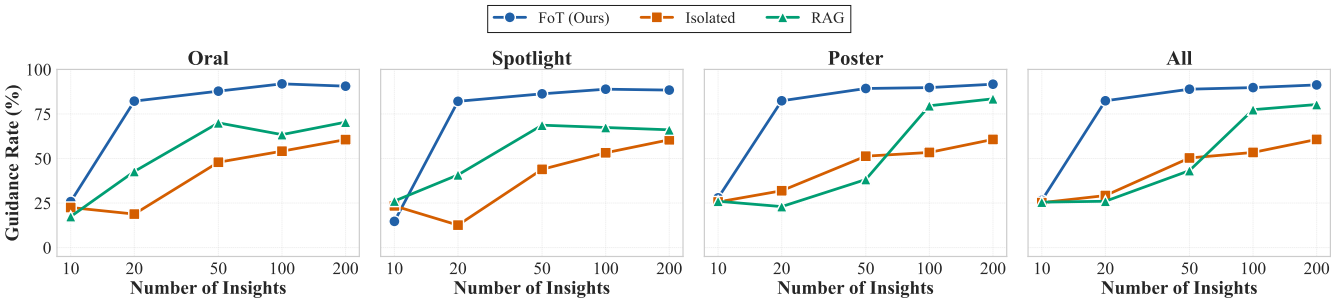

Figure 9. Comparison of FoT, isolated agents, and RAG in their ability to generate insights that can cover the core contributions of ICLR 2025 papers (guidance rate) under varying numbers of generated insights. Local agents use Gemini-2.0-Flash as the base LLM and are fed with ICLR 2024 papers as input. FoT generates higher-quality insights that better guide future research directions while requiring less input text, hence reducing token consumption.

Table 25. Accuracy on LiveMathBench Hard 2025 where there are four agents in total and each agent uses a different model. We can see that FoT can bring improvements over isolated agents. The server agent uses Gemini 3.1 Pro.

Algorithms	Gemini 3.1 Pro	Gemini 2.5 Flash	Gemini 2.5 Flash Lite	Gemini 2.5 Pro
Isolated Agents	0.720	0.420	0.360	0.400
FoT (Round 2)	0.710	0.420	0.400	0.320
FoT (Round 3)	0.720	**0.530**	**0.440**	**0.440**

previous papers), which takes more space than a lightweight insight library.

F.4 Effects of Agent Participation

We further analyze the impact of varying the number of participating agents in FoT under two settings. In Figure 10, each agent reads one paper while we vary the total number of papers, simulating partial participation as in cross-device federated learning. For each setting with 25%, 50%, and 75% local agents, we repeat the experiment three times, each with randomly selected papers to mitigate variation in paper quality. In Figure 11, we fix the total number of papers across agents but vary how many papers each agent reads before generating reasoning traces. When we use 2,260 agents, each agent generates reasoning traces after reading a single paper, whereas with 91 agents, each agent generates reasoning traces after reading 25 papers. We set the max insight library size to 200. For ICLR 2024 papers, we partition them across agents via primary research area so that each agent takes as input papers from the same areas. We observe that the number of agents does not significantly impact performance. However, assigning all papers to a single agent can result in

an input size that exceeds the model's context window when performing local reasoning. Table 26 and Table 27 present the performance on each benchmark for the two experimental settings in Table 7. In Table 26, all results except those on the diagonal from the top-left to the bottom-right correspond to evaluations where the benchmark is not used during insight library construction, and performance improvements are observed. In Table 27, the diagonal entries represent cases where the benchmark is not used during insight library construction, and improvements are also observed. These results confirm that, in FoT, the insight library can be effectively used by newly participating agents, as well as by agents that did not contribute to its construction, to improve performance.

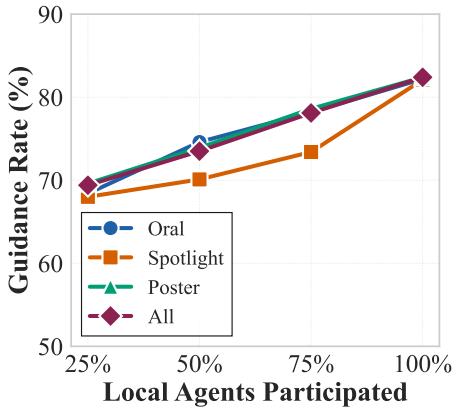

Figure 10. Paper guidance rate across different levels of local agent participation where each agent reads one paper. Guidance rate increases as participation increases (i.e., more input tasks).

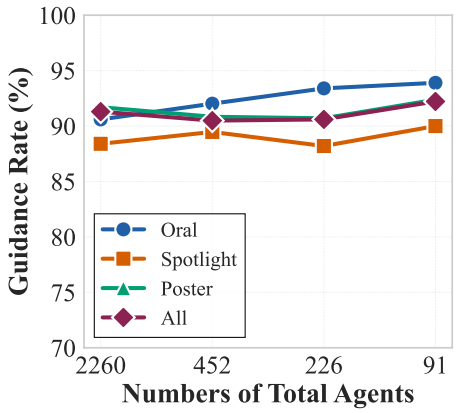

Figure 11. Paper guidance rate across different numbers of total agents, where the total number of papers read is fixed at 2260. The guidance rates remain high under different numbers of agents.

Table 28 compares the costs of the two settings with those of FoT and isolated agents. The costs of FoT and the two settings include agent inference, reasoning trace extraction, and server-side aggregation. In the Include-One setting, creating an insight library requires only about $0.15 in additional cost. Once constructed, this library can also benefit newly participating agents, improving their performance while reducing inference token usage. Therefore, the benefit of the insight library is persistent: agents that did not participate in library construction can still leverage it to achieve higher accuracy, use fewer tokens, and reduce long-term costs.

Table 26. Mathematical problem-solving performance on each benchmark for the 'Include One' experiment in Table 7. Improvements over the single-agent baseline are bolded.

Benchmark participating agent works on	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	Avg
AIME24	0.967	**0.967**	0.935	0.864	**0.944**	0.727	**0.810**	0.690	**0.863**
AIME25	0.967	0.933	0.935	0.864	**0.944**	0.727	**0.857**	0.690	**0.865**
AMC	**1.000**	0.933	0.935	0.864	0.889	0.727	**0.857**	0.680	**0.861**
CCEE	**1.000**	**0.967**	0.935	**0.909**	**0.944**	0.727	**0.857**	**0.700**	**0.880**
CNMO	0.967	0.933	0.935	0.864	0.889	0.727	**0.810**	**0.700**	**0.853**
WLPMC	0.967	0.933	0.935	0.864	0.889	0.727	**0.857**	0.690	**0.858**
V202412 Hard	0.967	0.933	0.935	0.864	0.889	0.727	**0.857**	**0.700**	**0.859**
V202505 Hard	**1.000**	0.933	**1.000**	**0.909**	**0.944**	**0.818**	**0.857**	**0.720**	**0.898**

Table 27. Mathematical problem-solving performance on each benchmark for the 'Exclude One' experiment in Table 7. Improvements over the single-agent baseline are bolded.

Benchmark non-participating agent works on	AIME24	AIME25	AMC	CCEE	CNMO	WLPMC	V202412 Hard	V202505 Hard	Avg
AIME24	0.967	**1.000**	0.913	0.864	0.889	**0.818**	**0.905**	**0.720**	**0.885**
AIME25	0.967	0.967	0.913	**0.909**	**0.944**	**0.818**	**0.857**	0.690	**0.883**
AMC	**1.000**	**0.967**	0.935	**0.909**	**1.000**	0.727	**0.905**	0.690	**0.892**
CCEE	0.967	0.933	0.935	**0.886**	**1.000**	**0.818**	**0.857**	0.680	**0.885**
CNMO	0.967	0.933	0.913	**0.909**	0.889	0.727	**0.857**	0.690	**0.861**
WLPMC	0.967	**0.967**	0.935	**0.909**	**0.944**	**0.818**	**0.857**	0.680	**0.885**
V202412 Hard	**1.000**	**0.967**	0.935	**0.909**	**0.944**	0.727	**0.857**	0.690	**0.879**
V202505 Hard	0.967	0.933	0.913	**0.886**	0.889	0.727	**0.857**	**0.710**	**0.860**

Table 28. Changes in average costs relative to the Isolated Agents baseline and FoT where all agents participate. Each row corresponds to one agent, identified by the benchmark it works on. **Include One**: The insight library is built from this agent alone. **Exclude One**: The insight library is built from all agents except this one.

	Include One		Exclude One	
Agent	Δ Isolated Agents	Δ FoT	Δ Isolated Agents	Δ FoT
AIME24	+0.15	−2.57	+1.15	−1.57
AIME25	+0.14	−2.58	+1.16	−1.56
AMC	+0.15	−2.57	+1.26	−1.46
CCEE	+0.15	−2.57	+1.23	−1.49
CNMO	+0.13	−2.59	+1.04	−1.68
WLPMC	+0.13	−2.59	+1.07	−1.64
V202412 Hard	+0.25	−2.47	+1.81	−0.91
V202505 Hard	+0.12	−2.60	+0.99	−1.73

G Complete Prompts

G.1 Prompts for Our Instantiation of FoT

Prompt 1 shows the prompt used by agents when an insight library is available. In Sections 4.1, 4.2 and 4.4, we directly apply this prompt to generate responses. In Section 4.3, we incorporate this prompt into the OpenClaw system prompt by specifying that "available insights are stored in `insight.md`." When no insight library is provided (i.e., isolated agents), the prompt reduces to "Problem: {problem}," where {problem} corresponds to the original dataset query. We put each problem prompt in Appendix G.2.

Prompt 1. Generating solution

```
Available Insights to Guide Your Solution:

{insight library}

---
INSTRUCTIONS: Review the insights above and
    actively apply the relevant techniques
    from insights to solve this problem.
    Consider which insights can help you
    approach the problem more effectively.

Problem: {problem}
```

Prompt 2 is used for solution reflection, which is the first part of 'Step 2: Local reasoning and self-reflection' (Figure 1), before an agent generates reasoning traces.

Prompt 2. Reflecting on Solution

```
Analyze the solution below to extract
    procedural knowledge that reflects the
    reasoning traces.

Problem:
{problem}

Step-by-Step Solution:
{solution}

Your task: Extract the fundamental
    techniques used in resolution that can
    be packaged as reasoning traces. Focus
    on:

1. What step-by-step procedures were used?
    How can these be repeated?
2. What conditions determined which
    approach to use? When should each
    technique apply?
3. What methods, strategies, or workflows
    can be applied to similar problems?
4. What made this approach effective? What
    should someone know to use it correctly?
5. What types of problems would benefit
    from these techniques?

Output your analysis covering:

### I. Procedural Knowledge
- Break down the solution into clear,
    repeatable procedures
- The extracted traces should be concrete
    solutions rather than general
    principles.

### II. Reusable Techniques and Methods
- List specific techniques, strategies, or
    workflows used
- The techniques should be solid on
    practical questions rather than very
    general and high-level principles.
- For each technique, identify:
  * When it should be used
    (conditions/triggers)
  * How it was applied (concrete steps)
  * Why it was effective (insights)
  * What problems it could solve
    (applicability)

### III. Critical Insights and Guidelines
- What key insights made this solution work?
- What common pitfalls should be avoided?
- What variations or edge cases should be
    considered?

Focus on extracting actionable, procedural
    knowledge that can be packaged as
    reusable insights for similar problems.
```

Prompt 3 is used for the generation of reasoning traces, which is the second part of 'Step 2: Local reasoning and self-reflection' (Figure 1). The description includes our definitions of reasoning traces.

Prompt 3. Generating Reasoning Trace

```
Extract reasoning traces from the solution
    below. Analyze the solution and
    reflection to identify concrete,
    actionable traces so that similar
    problems can be solved via the traces

Problem: {problem}

Solution: {solution}
```

```
Reflection: {reflection}

**Your Task:**
Identify and extract all reusable reasoning
    traces, techniques, and methods used in
    the solution. Each trace should be a
    concrete procedure that can guide
    someone to solve similar problems.

**What Makes a Good Reasoning Trace:**
- A specific technique or method that was
    used in the solution.
- Something that can be applied to similar
    problems, not just this one.
- Includes guidance on when and how to use
    it with clear steps that can be
    followed if necessary.
- Not repetition of already well-known or
    commonly adopted techniques.
- Not too general and high-level but
    contains actionable procedural
    knowledge.

**Description Must Include:**
1. **Core idea**: The fundamental concept
    of what this trace is about. What is
    the main technique or method? What does
    it do?

2. **When to use**: Explain when this skill
    should be applied. What types of
    problems? What conditions must be met?
    What situations trigger this skill?

**Output Format (Simple JSON):**
Output a simple JSON object with skill
    names as keys and descriptions as
    string values:

{{"trace_name": "description"}}

Format Rules:
- Use valid JSON format
- Each trace name must start with "trace_"
- Keep JSON simple - no nested objects,
    just key-value pairs
- Escape quotes in descriptions with
    backslash: \\"

**Example:**
{{
  "trace polynomialFactoring": "The major
    idea is how we can turn a polynomial
    into a product of simpler expressions.
    This skill is particularly useful for
    quadratic and higher-degree polynomial
    equations where factoring can simplify
    the problem. Factoring reduces complex
    polynomials to simpler equations. When
    solving equations with polynomial
    expressions that can be factored,
    especially when the polynomial has
    recognizable patterns like difference
    of squares (a^2-b^2), perfect square
    trinomials (a^2+2ab+b^2), or common
    factors.
  "trace depthFirstSearchImplementation":
    "This algorithm is essential for
    problems involving path finding, cycle
    detection, topological sorting,
    connected components, or exploring all
    possible solutions in a search space.
    DFS explores depth before breadth,
    using stack-based recursion or explicit
    stack. It is memory-efficient for deep
    structures and naturally handles
    backtracking. The visited set prevents
    infinite loops and redundant work. DFS
    is the foundation for many graph
    algorithms including topological sort,
    strongly connected components, and maze
    solving. When you need to explore or
    traverse a graph, tree, or nested
    structure systematically, going as deep
    as possible before backtracking. Use
    DFS when you need to visit all nodes in
    a connected component, find paths
    between nodes, detect cycles, or
    explore recursive structures like file
    systems, nested data, or game states. "
}}

**Output your response as a valid JSON
    object only:**
```

Prompt 4 is used for clustering and identifying connections between reasoning traces, as part of 'Step 4: Aggregate traces to update the library' in Figure 1, before updating the insight library.

Prompt 4. Constructing Relationship between Reasoning Traces

```
You are analyzing a collection of reasoning
    traces generated for problem-solving.
    Understand their relationships and
    structure and build a profile of their
    relationships.

**Collected {len(insight_store)} traces in
    total:**
Note: This collection includes ALL traces
    from all problems without
    deduplication. Similar names with
    different indices (e.g., _001, _002)
    represent different occurrences that
    may have variations in their
    descriptions.

{insights_text}

**Your Task:**
Analyze these traces and build a profile of
    their relationships:

1. **Identify Clusters**:
   Group related traces that share:
   - Resolve the same or similar problem
   - Similar approaches or techniques
   - Nearly identical traces (e.g., same
    trace with minor variations in
    description or parameters)
   - Traces in the same cluster should be
    highly similar.

2. **Build Trace Relationships**:
Record all important relationships - traces
    don't exist in isolation and build a
    relationship graph that records:
   - **Prerequisite relationships**: Traces
    that must be learned/used before others
   - **Composition relationships**: Traces
    that can be chained/composed together
   - **Alternative relationships**:
    Different approaches to the same problem
   - **Complementary relationships**:
    Traces that work better together than
    individually used
   - **Derivation relationships**: Traces
    derived from or based on others
   - **Similar relationships**: Traces that
    are similar but not identical
Map relationships between traces within
    clusters and across clusters.

# Output Format:
{{
  "clusters": [
    {{
      "cluster_id": 0,
      "cluster_name": "Domain/Theme Name",
      "traces": ["name1", "name2", "name3"],
      "theme": "What is the high-level
    technical idea of the traces in this
    cluster?",
    }}
  ],
  "relationships": [
    {{
      "trace_a": "trace_name1",
      "trace_b": "trace_name2",
      "relationship_type":
    "prerequisite/complementary/alternative/similar/der
      composes_with",
      "description": "How these traces
    relate to each other and Why"
    }}
  ]
}}

**Output your analysis as JSON only:**
```

The output relationship dictionary from Prompt 4 is double-checked and formatted as valid JSON to facilitate downstream extraction and processing. Prompt 5 is used for updating the insight library, as the second part of 'Step 4: Aggregate traces to update the library' in Figure 1.

Prompt 5. Building Insight Library

```
**Your Task:**
You are extracting fundamental insights
    from a collection of problem-solving
    traces.

**Output Requirements (STRICT):**
1. Return EXACTLY one valid JSON object and
    nothing else.
2. Do NOT output markdown code fences.
3. Do NOT output explanations, notes,
    reasoning, prefixes, suffixes, or
    `<think>` content.
4. Do NOT output list/array at top-level.
5. Every key must start with "insight_".
6. Every value must be a single string.
7. No nested objects, no nested arrays.

**Required JSON shape:**
{{
    "insight_name1": "description string",
    "insight_name2": "description string"
}}

**Formatting Rules:**
- Use valid JSON syntax only.
- Keep top-level as key-value pairs only.
```

```
- Escape quotes in descriptions with
    backslash: \\"
- If uncertain, still output a valid JSON
    object (possibly with fewer insights),
    never free text.

Your goal is to extract a comprehensive set
    of fundamental, cross-domain insights
    that can be derived and applied beyond
    their original domain and meet the
    following requirements:
- Combine previous insights (if any):
    {existing_encyclopedia if
    existing_encyclopedia else "None"} with
    new insights.
- Extract your insights based on all client
    reasoning traces: {all_insights_text}.
    These traces are derived from solving
    specific problems (bottom-up approach)
- Use clusters of reasoning traces:
    {clusters_text if clusters_text else
    "None identified"} to help organize.
- Use relationships between traces:
    {json.dumps(profiling.get('relationships',
    []), indent=2) if isinstance(profiling,
    dict) else "None identified"} to help
    organization
- Your task is to extract
    multi-disciplinary, fundamental
    knowledge (top-down approach) which can
    be generalized to multi-domain
    problem-solving.
- The extracted insights should be able to
    DERIVE and GUIDE the use of the
    collected insights
- The extracted insights cannot be too
    general. They are not supposed to be
    knowledge which can be applied to any
    problem. They should be fundamental
    knowledge to particular several domains
    but specific.
- You should extract proper number of
    insights. Not too few. Not too many. Do
    not over simplify or be too detailed.
- DO NOT over-merge insights.

Insights should have following properties:
1. **Extract Reusable Primitives**:
   - For EACH cluster, extract multiple
    fundamental insights capturing core
    essence and variations (DO NOT
    over-merge)
   - Identify cross-domain patterns that
    apply to multiple fields
   - Create reusable, composable primitives
    specific enough to be actionable

2. **Knowledge to include**:
   - **Fundamental Level**: Core principles
    underlying multiple domains
   - **General Level**: Broad techniques
    for related problem types
   - **Cross-Domain**: Insights
    transferable beyond origin field

3. **Preserve While Generalizing**:
   - Create fundamental versions that can
    guide/derive original insights
   - Maintain important variations rather
    than collapsing into single insight

4. **Description Format:**
   Each description is a single string
    containing:
   - What the insight is and how it solves
    problems
   - When to use: problem types,
    conditions, triggers (be comprehensive
    and specific)

**Example 1 - Transformer Architecture:**

Input reasoning traces:
```

```
- reasoning trace
    VisionTransformerImageClassification:
    "I need to classify medical X-ray
    images into disease categories. CNNs
    aren't working well - they can't
    capture relationships between distant
    regions like how fluid in the lower
    right lung might relate to heart
    enlargement. Let me try Vision
    Transformer (ViT). I'll divide each
    X-ray into 16x16 patches - a 224x224
    image gives 196 patches. Each patch
    becomes like a token in NLP. I flatten
    each to a 256-dim vector. Since
    transformers don't know spatial
    positions, I add position embeddings so
    the model knows patch [0,0] is
    top-left. I prepend a [CLS] token to
    gather global info. Feeding through 12
    transformer encoder layers - the
    self-attention lets every patch attend
    to every other patch directly, so patch
    [2,5] can look at patch [10,12] even
    though they're far apart spatially.
    This is exactly what I need! After 12
    layers, I extract the [CLS] token and
    feed to MLP classifier. Training on 50k
    chest X-rays: 94.2% accuracy, beating
    ResNet-50's 89.3%. The attention maps
    show it's correctly attending to both
    lungs simultaneously for pneumonia
    detection, linking heart size to lung
    fluid - this cross-region reasoning is
    what CNNs miss. The key: treating image
    patches as tokens with self-attention
    enables global spatial reasoning."

- reasoning trace
    TransformerNextWordPrediction: "I'm
    building autocomplete for a text
    editor. The challenge: predict next
    word given arbitrary-length context.
    RNNs struggle with long sequences - the
    hidden state forgets earlier context.
    Let me use a transformer decoder. I
    tokenize 'The cat sat on the' using
    WordPiece -> tokens [254, 8901, 4523,
    651, 278]. Convert each to 512-dim
    embedding and add positional encodings.
    Critical part: causal masking so the
    model can't cheat by seeing future
    words. When predicting token at
    position 4, it should only see 0-3. I
    implement lower-triangular attention
    mask. Processing through 6 decoder
    layers with masked self-attention. At
    position 4 ('the'), attention computes
    similarity between its query and keys
    of previous words. It attends strongly
    to 'sat' (0.8) and 'on' (0.7), weakly
    to 'cat' (0.2). Using 12 heads helps -
    different heads capture different
    patterns: head-2 learns syntax
    (preposition+article), head-5 learns
    semantics (actions+objects), head-8
    learns long-range dependencies. After
    final layer, project last hidden state
    through 50k-dim softmax. For 'the': top
    predictions are 'mat' (0.73), 'floor'
    (0.12), 'rug' (0.08). Deployed in
    production - users accept top-3
    suggestion 85\% of the time, reducing
    typing by 40\%. Transformer's
    self-attention captures context way
    better than RNN's sequential
    processing."

Output aggregated insight:
{{
```

```
  "insight_transformerArchitecture": "This
    fundamental neural network architecture
    applies across natural language
    processing, computer vision, time
    series analysis, graph neural networks,
    and multi-modal learning. This insight
    is essential for modern AI applications
    including language models, image
    processing, code generation, and
    scientific computing. When you need to
    capture relationships between all
    elements simultaneously
    (self-attention), you're working with
    sequences of variable length, you need
    parallel processing of sequences, or
    when the problem involves understanding
    context and relationships. Details: 1)
    Design input representation - convert
    your data into embeddings (token
    embeddings for text, patch embeddings
    for images, node embeddings for
    graphs), add positional encodings to
    preserve sequence information, and
    prepare input for transformer blocks 2)
    Create models with transformer blocks
    3) Apply task-specific architecture -
    use encoder-only for understanding
    (BERT, ViT), decoder-only for
    generation (GPT), or encoder-decoder
    for translation"
}}

**Example 2 - Surface-Enhanced Raman
    Spectroscopy:**

Input reasoning traces:

- reasoning trace SERSMedicalDetectionR6G:
    "I need to detect cancer biomarkers in
    blood at incredibly low concentrations
    - 10^-12 M, like finding molecules in a
    swimming pool. ELISA only goes to 10^-9
    M, not sensitive enough for early
    diagnosis. Let me try SERS -
    Surface-Enhanced Raman Spectroscopy.
    Metal nanoparticles create huge EM
    field enhancements. I synthesize 60nm
    gold nanoparticles via citrate
    reduction. At 785nm laser, these have
    plasmon resonance amplifying local
    field by ~10^6. But I need selectivity
    too - can't detect everything. So I
    functionalize the gold with anti-PSA
    antibodies for prostate cancer. When I
    add patient serum, only PSA proteins
    bind. Here's the clever part: I add R6G
    (Rhodamine 6G) reporter molecules. R6G
    has enormous Raman cross-section and
    when it sits in nanogaps between gold
    particles, field enhancement shoots to
    10^8 or 10^10. Incubate 30 min for PSA
    binding, add R6G which sticks near
    bound PSA. Hit with 785nm laser at 5mW
    - I see characteristic R6G peaks at
    1650, 1510, 1310 cm^-1. Peak intensity
    directly proportional to PSA amount.
    Integrate 60 sec for good SNR.
    Comparing to calibration: detecting PSA
    at 0.1 ng/mL - that's 10,000x more
    sensitive than ELISA! On clinical
    samples, detected prostate cancer 3-6
    months earlier than conventional tests.
    The breakthrough: combining selective
    antibody recognition with SERS
    amplification gives single-molecule
    sensitivity while maintaining
    specificity."
```

```
- reasoning trace SERSPollutantDetection:
    "Monitoring river water for pesticides.
    EPA limit for malathion is 0.1 ppb but
    standard chromatography needs 1 ppb
    minimum. I need 10x better for early
    warning. SERS might work. Instead of
    spherical particles, I'll fabricate
    silver nanorod arrays - sharp tips
    create hotter hotspots than spheres.
    Using oblique angle deposition: 80nm
    nanorods with 4:1 aspect ratio on
    silicon. Gaps between rods only 5-10nm
    - perfect for trapping molecules. I
    calculate enhancement should hit 10^10
    at 532nm. Collect river water, filter
    through 0.2\mu m to remove debris and
    bacteria. Drop 50 \mu L onto nanorod
    substrate. During 5-min adsorption,
    pesticide molecules diffuse into
    nanogaps. Small gap means molecules
    guaranteed in enhancement zone (<10nm
    from metal). Rinse gently - removes
    interfering organics/salts but leaves
    adsorbed pesticides. Excite with 532nm
    at 2mW, matching silver plasmon peak.
    Even at 0.01 ppb malathion, clear peaks
    at 1440 cm^-1 (P=S stretch), 1080 cm^-1
    (P-O-C), 640 cm^-1 (C-S). Measuring
    1440 peak height vs calibration
    standards for quantification. Tested 50
    river sites, cross-validated against
    LC-MS: R^2=0.97 correlation. Best part:
    do this in field with portable Raman -
    no lab needed. Real-time monitoring at
    10x below regulatory limits. The
    nanorod geometry is critical - those
    sharp tips and tight gaps push
    enhancement to 10^10."

Output aggregated insight:
{{
    "insight_surfaceEnhancedRamanSpectroscopy":
    "This powerful technique applies across
    analytical chemistry, materials
    science, biosensing, pharmaceutical
    analysis, environmental monitoring, and
    forensics. The technique achieves
    single-molecule sensitivity (10^6-10^11
    enhancement) while providing molecular
    structural information through
    vibrational fingerprints. When to use:
    When you need ultra-sensitive detection
    below conventional analytical limits,
    when you want label-free molecular
    identification, when analyzing trace
    contaminants or biomarkers, or when
    field-portable real-time analysis is
    required. Common steps: 1) Prepare
    SERS-active substrate - synthesize
    plasmonic nanostructures (gold/silver
    nanoparticles, nanorods, nanostars)
    optimizing particle size (20-100 nm),
    shape, and inter-particle spacing (1-10
    nm gaps) to maximize electromagnetic
    field enhancement at laser wavelength
    2) Functionalize substrate if needed -
    modify metallic surface with
    antibodies, aptamers, or molecular
    recognition elements for selective
    analyte binding and improved
    specificity 3) Prepare and apply sample
    - process sample (filter, dilute,
    concentrate as needed), deposit onto
    SERS substrate via drop-casting or
    flow-through, allow adsorption time for
    molecules to enter hot spots (<10 nm
    from metal surface) 4) Select laser
    parameters - choose wavelength matching
    plasmon resonance (532, 633, or 785
    nm), optimize power (0.1-10 mW) to
    avoid sample damage while maximizing
    signal 5) Acquire SERS spectrum -
    collect Raman scattered light with
    appropriate integration time, record
    vibrational spectrum showing
    characteristic molecular peaks 6)
    Analyze spectral fingerprint - identify
    molecules by comparing peak positions
    to reference spectra, quantify
    concentration from peak intensities
    using calibration curves, assess
    molecular orientation from peak ratios
    7) Validate and control quality -
    average multiple spots for
    reproducibility, use internal
    standards, verify with orthogonal
    methods, consider substrate
    heterogeneity and enhancement factor
    variations"
```

```
}}
```

Prompt 6 is used for determining whether a paper can be covered by insights from the library, used in the application studied in Section 4.4.

Prompt 6. Judging Whether Paper Contributions Match with the Insight Library

```
You are evaluating whether a research
    paper's proposed solutions are guided
    by or directly derived from insights in
    an encyclopedia.

Insights:
{insights_prompt}

Research Paper:
{paper_text}

Evaluation Criteria - An insight guides the
    paper ONLY IF ALL of the following are
    true:

1. CONCRETE METHODOLOGY USAGE: The
    insight's methodology or approach is
    concretely used in the paper's
    methods/approach section, not just
    theoretically relevant or mentioned in
    motivation.

2. METHODS SECTION PRESENCE: The insight
    must be related to how the paper
    actually implements its solution
    (methods, algorithms, techniques), not
    just in problem statement or related
    work.

3. COUNTERFACTUAL TEST: The paper's core
    contribution would fundamentally differ
    or fail without this insight. Ask: "If
    the authors didn't know this insight,
    could they still arrive at the same
    core solution?"

4. SPECIFICITY: The insight must
    specifically address a key challenge or
    component of the paper's solution, not
    just be generally applicable background
    knowledge.

Response Format:
- Respond ONLY in valid JSON with keys:
    guided (boolean), matched_insights
    (array of insight names)
- Set guided=true ONLY when at least one
    insight passes ALL criteria above
- Use only exact insight names from the
    Insights list above
```

G.2 Complete Problem Prompts

For completeness, we summarize the problem prompts used in each application. For math benchmarks, we concatenate the dataset question with Prompt 7. For GPQA and GPQA-diamond, we use Prompt 8; for LiveCodeBench, Prompt 9; and for Humanity's Last Exam, Prompt 10. For the research insight library, we adopt the template in Prompt 11, using the answer as the "{solution}." For PinchBench and Claw-Eval, we directly use the task instructions provided by the benchmarks.

Prompt 7. Math Benchmark

```
'\nSolve the problem step by step. Wrap
    your final answer in "\\boxed{}".'
```

Prompt 8. Scientific Question Answering Benchmark

```
Please solve this graduate-level science
    question step by step.

Instructions:
1. Analyze the question carefully
2. Consider each option systematically
3. Explain your reasoning
4. Provide your final answer as a single
    letter (A, B, C, or D)
5. Wrap your final answer letter in
    \\boxed{}, for example: \\boxed{A}
```

Prompt 9. Coding Benchmark

```
Please solve this programming problem by
    writing clean, efficient code.

Requirements:
1. Read the problem description carefully
2. Understand the input/output format
3. Write a complete solution
4. Include proper input/output handling
5. Wrap your final code solution in
    markdown code blocks with triple
    backticks (```)
```

```
Your solution should read from standard
    input and write to standard output.
```

Prompt 10. Humanity's Last Exam

```
You are solving one HLE multimodal
    benchmark problem.
Use BOTH the question text and the attached
    image.
Return only the final answer, with no
    explanation.
```

Prompt 11. Paper Reading

```
Answer research question of the paper:
    {paper_name} based on paper
    content:{paper_content}
All papers need to be considered in the
    analysis.
```

G.3 Prompts for Metacognitive Reuse [23]

For completeness, here, we re-state the prompts proposed in prior work [23] that we build upon and compare against.

Prompt 12. Generating Solution

```
Please reason step by step and put the
    final answer in
Problem: <problem>
```

Prompt 13. Generating Reflection

```
Here is the definition of a behavior:
- A behavior is a note or skill to keep in
    mind while solving math problems.
- It can be a strategy, a trick, or a
    technique.
- It can also be a general rule or a common
    sense principle.
- A behavior is not a solution to the
    problem, but it can be used to solve
    the problem.

For example - if the problem is "Find the
    area of a circle with radius 4", one
    useful behaviour could be
{"behavior_area_of_circle": area of a
    circle is pi*r^2}.

Given a problem and the corresponding
    solution, react and critique the
    solutions along the following
    dimensions:

1. Correctness Analysis: Is the answer
    mathematically correct? Are there
    calculation errors? Is the reasoning
    logically
sound? Are all steps properly justified?
    What specific mistakes were made?
2. Missing Behaviors Analysis: What
    behaviors should have been used but
    weren't? Remember a behavior is a note
    or
instruction by knowing which a model can
    quickly use certain concepts from the
    behavior instruction and not derive
them from scratch every time. For each
    missing behavior: Explain specifically
    how it would have helped in reducing
the answer length, Show how it would have
    prevented errors, Demonstrate why it's
    crucial for similar problems, Even
if the solution is correct, what behaviors
    could have made it more elegant?
3. New Behavior Suggestions: Suggest
    specific new behaviors that will help
    with similar problems. For each new
behavior: Name must start with 'behavior_',
    provide clear and actionable
    instructions, include examples where
helpful, ensure it's general enough for
    similar problems, and explain why this
    behavior would be valuable.
```

Prompt 14. Generating Behavior

```
<reflection prompt>

<reflection>

Now, given this reflection generate a list
    of
behaviors and corresponding instructions
    explanations in json format. Each
    behavior should be a single line, and
    the format is "behavior_[name]:
    [description]". The list should
```

```
be in json format, and each behavior should
    be a key-value pair, where the key is
    the behavior name and the value is the
    description.
```

H Insights, Raw Reasoning Processes, and Reasoning Traces

H.1 Representative Insights

In this section, we show example insights. The first three examples are from the library that is created in the ML research insight discovery application, derived from ICLR 2023 papers (Setting 1).

With Gemini 3.0 Pro Preview and ICLR 2023 papers, we obtain three insights: structured output prediction (Insight 1), causal representation learning (Insight 2), and physics-informed deep learning (Insight 3), which provide interesting ideas and instructions on how to approach related research problems. These topics remain cutting-edge research questions today.

Insight 1. Structured Output Prediction (Setting 1)

```
When to use: When the model output is not a
    simple class or scalar, but a complex
    structured object (e.g., a sequence, a
    tree, a matching, a code program) where
    dependencies between output elements
    must be modeled. Step-by-step: 1)
    Define the output space constraints and
    dependencies (e.g., grammar rules,
    bi-partite matching constraints). 2)
    Use autoregressive models (like
    Transformers) to generate elements
    sequentially conditioned on history, or
    non-autoregressive models with
    iterative refinement. 3) Apply
    structural losses (e.g., connectionist
    temporal classification, graph edit
    distance, optimal transport cost)
    rather than simple pointwise error. 4)
    Use inference techniques like beam
    search or constrained decoding to find
    the most likely valid structure. Key
    insights: This covers skills like
    hierarchical bipartite matching,
    permutation equivalent modeling, and
    program synthesis.
```

Insight 2. Causal Representation Learning (Setting 1)

```
When to use: When building models that must
    robustly identify cause-and-effect
    relationships rather than spurious
    correlations, enabling interventions
    and counterfactual reasoning.
    Step-by-step: 1) Model the system using
    a Structural Causal Model (SCM) or
    Directed Acyclic Graph (DAG). 2)
    Distinguish between invariant causal
    mechanisms (stable across environments)
    and spurious associations. 3) Use
    interventions (do-calculus) or
    environmental diversity to discover the
    causal graph. 4) Learn representations
    that satisfy conditional independence
    constraints implied by the causal
    structure (disentanglement). 5)
    Optimize for Invariant Risk
    Minimization (IRM) to find predictors
    that are optimal across all
    interventional distributions. Key
    insights: This unifies skills like
    causal mediation analysis, front-door
    prompting, and causal shift diagnosis.
```

Insight 3. Physics Informed Deep Learning (Setting 1)

```
When to use: When modeling physical systems
    governed by partial differential
    equations (PDEs), conservation laws, or
    geometric constraints (e.g., fluid
    dynamics, structural mechanics, climate
    modeling) where purely data-driven
    models violate physical validity or
    require excessive data. Use this to
    integrate physical knowledge directly
    into the learning process.
    Step-by-step: 1) Formulate the
    governing physical equations
    (residuals) of the system (e.g.,
    Navier-Stokes, Eikonal equation). 2)
    Design the neural network to output the
    physical state variables (velocity,
    pressure) given spatiotemporal
    coordinates. 3) Compute derivatives of
    the outputs with respect to inputs
    using automatic differentiation
    (Autograd) to evaluate the PDE
    residuals. 4) Construct a composite
    loss function: Data Loss (fitting
    observations) + Physics Loss
    (minimizing PDE residuals) +
    Boundary/Initial Condition Loss. 5)
    Optionally, use hard constraints
    (ansatz) or coordinate transformations
    to enforce boundary conditions by
    construction. Key insights: This guides
    specific skills like PDE-constrained
    latent dynamics, soft contact modeling,
    and neural operator learning.
```

In Setting 2, insights generated by Gemini 2.0 Flash are generally shorter than those by Gemini 3.0 Pro. However, even without knowledge of future papers, we still obtain interesting insights such as neuro-symbolic reasoning (Insight 4) and parameter-efficient model adaptation (Insight 5), which corresponds to what is currently known as parameter-efficient fine-tuning. It also produces the zero-to-one capability testing insight (Insight 6), which can be adapted to systems research.

Insight 4. Neuro Symbolic Reasoning (Setting 2)

```
When to use: When a problem requires both
    the perceptual flexibility of neural
    networks (handling noisy, unstructured
    data) and the
    precision/interpretability of symbolic
    logic (handling rules, math,
    constraints). Step-by-step: 1)
    Decompose the system into a Neural
    Module (perception/pattern matching)
    and a Symbolic Module
    (reasoning/execution). 2) Design an
    interface layer: converting neural
    outputs into discrete symbols
    (concepts) or differentiable
    approximations (soft logic). 3) Use the
    Symbolic Module to execute logical
    rules, programs, or arithmetic on the
    extracted symbols. 4) Train end-to-end
    (via gradients through soft logic) or
    separately (via REINFORCE or
    supervision) to optimize both
    perception and reasoning accuracy. Key
    insights: This abstracts skills like
    neuro-symbolic prompt construction,
    logic constraint integration, and
    program synthesis.
```

Insight 5. Parameter Efficient Model Adaptation (Setting 2)

```
When to use: When adapting large
    pre-trained 'foundation' models (LLMs,
    ViTs) to specific downstream tasks
    without the computational cost of full
    fine-tuning. Step-by-step: 1) Freeze
    the bulk of the pre-trained model
    parameters. 2) Introduce a small set of
    trainable parameters: Adapters
    (bottleneck layers between existing
    layers), Prompt/Prefix Tuning
    (learnable input tokens), or Low-Rank
    Adaptation (LoRA - decomposing weight
    updates into low-rank matrices). 3)
    Train only these new parameters on the
    task-specific dataset. 4) Optionally,
    use techniques like weight averaging
    (model soup) or dynamic routing to
    combine multiple adaptations. Key
    insights: This covers skills like
    minimalist adapter co-training, prompt
    engineering, and efficient fine-tuning
    protocols.
```

Insight 6. Zero-To-One Capability Testing (Setting 2)

```
When to use: When assessing the functional
    value of a new technology or method to
    distinguish between efficiency gains
    and capability unlocks. This applies to
    product management, feature
    prioritization, and evaluating
    'breakthrough' research. Step-by-step:
    1) Ignore quantitative metrics (speed,
    cost, error rate) initially 2) Define
    the 'Capability Boundary' of the
    previous
    state-of-the-art---specifically what
    tasks it physically could not perform
    3) Ask the binary question: 'Does the
    new solution enable a task that was
    previously impossible/intractable?' 4)
    If the answer is 'No' (it just does
    existing tasks better), classify as
    '1-to-N' (Incremental) 5) If the answer
    is 'Yes' (it opens a new action space),
    classify as '0-to-1' (Non-Incremental)
    6) Prioritize 0-to-1 capabilities for
    strategic innovation as they create new
    markets.
```

The following examples are drawn from an insight library generated by Gemini-3.1-Pro under a high-thinking setting on PinchBench. These examples show that FoT can produce insights that serve as decision support in areas such as software engineering and market analysis.

Insight 7. Declarative Architecture Standardization (PinchBench)

```
This software engineering standard applies
    modern, declarative configurations and
    specific directory layouts to project
    initialization to ensure long-term
    maintainability and out-of-the-box
    compatibility with modern build tools.
    When to use: Whenever initiating,
    refactoring, or defining the
    architecture of new code libraries,
    packages, or microservices. Details: 1)
    Implement the 'src' layout paradigm
    (placing core modules inside a src/
    directory adjacent to tests) to force
    testing against the installed package
    version rather than local directory
    masking. 2) Utilize declarative
    metadata configuration files (like
    pyproject.toml compliant with PEP
    517/518) instead of legacy imperative
    scripts (like setup.py) to manage
    dependencies and build instructions.
```

Insight 8. Differentiator Focused Market Deconstruction (PinchBench)

```
This competitive intelligence framework
    distills crowded market landscapes into
    highly actionable artifacts by ignoring
    parity and focusing exclusively on
    divergence. When to use: When
    conducting tech stack evaluations,
    vendor selections, or business
    intelligence reporting in mature
    industries where competitors share a
    massive baseline of identical
    functionalities. Details: 1) Scan macro
    industry trends before drilling into
    micro profiles. 2) Explicitly identify
    and filter out industry-standard
    baseline features ('table stakes'). 3)
    Abstract highly volatile data (like
    exact dollar costs) into broader
    operational paradigms. 4) Integrate
    recent structural context like M&A
    parent-company ownership. 5) Map the
    remaining unique architectural roots
    and target audience differences onto a
    structured cross-dimensional comparison
    matrix.
```

Insight 9. UI Bypass File Routing (PinchBench)

```
This fundamental operational pattern
    separates back-end automated artifacts
    from front-end conversational
    interfaces. It solves the 'double
    print' token clutter problem by writing
    generated payloads directly to the file
    system while returning only a
    synthesized 'Bottom Line Up Front'
    (BLUF) or status summary to the user
    interface. When to use: Apply this
    whenever executing tasks that generate
    large artifacts (codebases, CSVs,
    long-form reports) or orchestrate
    closed-loop file transformations
    (read-process-write), specifically when
    the user provides explicit destination
    commands like 'Save to' or 'Create a
    file'. Details: 1) Generate the primary
    payload internally. 2) Pipe payload via
    a file-write tool directly to
    persistent storage. 3) Verify
    successful byte-count written. 4)
    Output only a brief executive summary
    confirming completion and detailing
    core contents.
```

H.2 Complete Raw Reasoning Process for Case Studies in Appendix D.2

Reasoning Process 1. Incorrect Reasoning Process of GPQA Diamond Problem 89 (Round 2)

```
Solution Response: **Step-by-Step
    Analysis:**

1.  **Analysis of Product 1 Formation:**
    *   Starting Material:
    1,3-dibromoadamantane.
    *   Reagents: KOH, 240C.
    *   This reaction is a known Grob-type
    fragmentation of the adamantane
    skeleton. The 1,3-functionalization
    allows for the cleavage of a C-C bond
    in the cage, leading to a bicyclic
    system.
    *   The spectral data for Product 1
    confirms this:
        *   IR 1720 cm$^{-1}$ indicates a
    ketone ($C=O$).
        *   ${}^1$H NMR 4.79 ppm (2H)
    indicates terminal vinylic protons
    ($=CH_2$), which typically appear in
    the 4.5-5.0 ppm range.
        *   The structure consistent with
    this data and the starting material is
    **3-methylenebicyclo[3.3.1]nonan-7-one**
    (or
    7-methylenebicyclo[3.3.1]nonan-3-one
    due to symmetry). This molecule has the
    formula $C_{10}H_{14}O$ and fits the
    unsaturation index.

2.  **Analysis of Product 2 Formation:**
    *   Reagents: Product 1 + Excess
    Aluminum Isopropoxide ($Al(OiPr)_3$) +
    Heat.
    *   This is the
    **Meerwein-Ponndorf-Verley (MPV)
    reduction**. It selectively reduces
    ketones to alcohols.
    *   **Stereochemistry:** The MPV
    reduction is reversible and, under
    heated conditions with excess reagent,
    typically yields the **thermodynamic
    product**.
    *   In the bicyclo[3.3.1]nonane system,
    substituents at positions 3 and 7 face
    severe transannular steric repulsion if
    they are in the **endo**
    (equatorial-like pointing inwards)
    position. Therefore, the stable
    conformer places bulky groups in the
    **exo** (axial-like pointing outwards)
    position.
    *   Consequently, the thermodynamic
    product of the reduction is the
    **exo-alcohol** (where the OH group is
    axial).
    *   Product 2 is
    **exo-3-methylenebicyclo[3.3.1]nonan-7-ol**.

3.  **Analysis of Product 3 Formation:**
    *   Reagents: Product 2 + Ozone
    ($O_3$), -78C, followed by DMS.
    *   This is an **ozonolysis** reaction
    with reductive workup. It cleaves the
    exocyclic carbon-carbon double bond
    ($=CH_2$) and converts it into a
    carbonyl group ($C=O$).
    *   The stereocenter at C7 (bearing the
    OH group) is unaffected.
    *   Product 3 is
    **exo-7-hydroxybicyclo[3.3.1]nonan-3-one**.

4.  **NMR Prediction for Product 3:**
    *   **Target Proton:** The question
    asks for the coupling pattern of the
    "most deshielded hydrogen atom"
    (excluding exchangeable OH).
```

```
    *   In Product 3, the proton on the
    carbon bearing the hydroxyl group
    (H-C7) is the most deshielded due to
    the electronegativity of oxygen
    (chemical shift typically 3.5 - 4.5
    ppm).
    *   **Stereochemistry of H-C7:** Since
    the hydroxyl group (OH) is in the **exo
    (axial)** position (thermodynamic
    product), the geminal hydrogen atom
    (H-C7) must be in the **endo
    (equatorial)** position.
    *   **Coupling Analysis:**
        *   The H-C7 (equatorial) proton is
    coupled to four vicinal protons on the
    adjacent bridge carbons (C6 and C8).
        *   Due to the molecular symmetry
    (plane passing through C3-C7-C9), the
    protons on C6 are equivalent to those
    on C8.
        *   The neighbors are: H-6ax,
    H-6eq, H-8ax, H-8eq.
        *   Coupling constants for an
    equatorial proton in a chair-like ring
    are generally small: $J_{eq-ax} \approx
    2-5$ Hz and $J_{eq-eq} \approx 2-5$ Hz.
        *   Because the coupling constants
    are small and of similar magnitude, the
    splitting pattern typically appears as
    a **pentet** (quintet) (representing
    overlap of couplings to 4 protons).
        *   (Note: If the H were axial, it
    would show large trans-diaxial
    couplings (~10-12 Hz) and small
    axial-equatorial couplings, resulting
    in a distinct triplet of triplets.
    Since we have the thermodynamic
    exo-alcohol, the H is equatorial,
    ruling out the triplet of triplets).

**Conclusion:** The most deshielded
    hydrogen is the equatorial H at C7,
    which couples to four neighbors with
    similar small coupling constants,
    producing a pentet.

\boxed{D}
```

Reasoning Process 2. Correct Reasoning Process of GPQA Diamond Problem 89 (Round 3)

```
Solution Response: **Step 1: Structure
    Elucidation of Product 1**
The starting material is
    1,3-dibromoadamantane. Heating this
    with excess KOH at high temperatures
    ($240^\circ\text{C}$) is a known
    fragmentation reaction involving the
    adamantane skeleton. The conditions
    facilitate a Grob-type fragmentation of
    the 1,3-functionalized system to form a
    bicyclic compound. The data provided
    for Product 1 supports this:
*   **IR 1720 cm$^{-1}$**: Indicates a
    ketone (C=O).
*   **$^{1}\text{H}$ NMR**: The signal at
    $\delta 4.79$ (2H) corresponds to an
    exocyclic methylene group
    ($=\text{CH}_2$). The remaining protons
    (12H) fit the bicyclic skeleton.
The structure formed is
    **7-methylenebicyclo[3.3.1]nonan-3-one**.
    This molecule fits the formula
    ($C_{10}H_{14}O$) and the spectral data.

**Step 2: Structure Elucidation of Product
    2**
Product 1 is heated with excess aluminum
    isopropoxide. These are the conditions
    for a **Meerwein-Ponndorf-Verley (MPV)
    reduction**.
*   The MPV reduction converts the ketone
    at C3 to a secondary alcohol.
*   Crucially, MPV reductions are
    reversible and thermodynamically
    controlled, yielding the more stable
    alcohol isomer.
*   In 3-substituted bicyclo[3.3.1]nonanes,
    the **exo-isomer** (substituent in the
    equatorial position) is
    thermodynamically more stable than the
    endo-isomer (substituent in the axial
    position). This is because the endo
    position suffers from severe
    transannular steric repulsion with the
    C7 position (even with the change in
    hybridization at C7, the exo-alcohol
    remains the preferred thermodynamic
    product in these systems).
*   Therefore, Product 2 is
    **exo-7-methylenebicyclo[3.3.1]nonan-3-ol**.
*   Stereochemistry: The hydroxyl group is
    **exo (equatorial)**. Consequently, the
    carbinol hydrogen atom (H-3) is **endo
    (axial)**.

**Step 3: Structure Elucidation of Product
    3**
Product 2 is subjected to ozonolysis
    ($O_3$, then $Me_2S$).
```

```
*   Ozonolysis cleaves the exocyclic
    carbon-carbon double bond
    ($=\text{CH}_2$) at C7 to form a ketone
    ($C=O$).
*   The alcohol at C3 is unaffected.
*   Product 3 is
    **exo-7-hydroxybicyclo[3.3.1]nonan-3-one**
    (also named
    exo-3-hydroxybicyclo[3.3.1]nonan-7-one
    due to symmetry).

**Step 4: Analysis of the Target Proton**
The question asks for the coupling pattern
    of the "most deshielded hydrogen atom"
    in Product 3.
*   The protons in the molecule are:
    *   Protons alpha to the ketone (C2/C4
    or C6/C8): $\delta \approx 2.4$ ppm.
    *   Bridgehead protons (C1/C5): $\delta
    \approx 2.0$ ppm.
    *   Bridge protons (C9): $\delta
    \approx 1.8$ ppm.
    *   Carbinol proton (H-3, alpha to
    -OH): $\delta \approx 3.8\text{--}4.2$
    ppm.
*   The **carbinol proton (H-3)** is the
    most deshielded non-exchangeable proton
    due to the electronegativity of the
    oxygen atom.

**Step 5: Determining the Coupling Pattern**
We have established that the hydroxyl group
    is **exo (equatorial)**, placing the
    target hydrogen atom (H-3) in the
    **axial** position of the
    cyclohexane-like chair ring.
*   **Neighbors**: The C3 carbon is flanked
    by two methylene groups (C2 and C4).
    Due to the plane of symmetry in the
    molecule, C2 and C4 are equivalent.
*   **Coupling Partners**: The axial H-3
    couples with:
    *   Two **axial** protons (H-2ax,
    H-4ax).
    *   Two **equatorial** protons (H-2eq,
    H-4eq).
*   **Coupling Constants ($J$)**:
    *   Axial-Axial coupling ($J_{aa}$) is
    large, typically 10-12 Hz
    (anti-periplanar arrangement).
    *   Axial-Equatorial coupling
    ($J_{ae}$) is small to medium,
    typically 2-5 Hz (gauche arrangement).
*   **Pattern**:
    *   The splitting by the two equivalent
    large couplings ($J_{aa}$) creates a
    triplet ($1:2:1$).
    *   Each line of this triplet is
    further split into a triplet by the two
    equivalent small couplings ($J_{ae}$).
    *   The resulting pattern is a large
    triplet of small triplets, known as a
    **triplet of triplets** (tt).

In contrast, an equatorial proton (which
    would result from the kinetic
    endo-alcohol) typically shows small
    couplings to all neighbors, appearing
    as a pentet or broad multiplet. Since
    we have the thermodynamic product
    (axial H), the answer is the triplet of
    triplets.

\boxed{C}
```

Reasoning Process 3. Incorrect Reasoning Process of AIME25 Problem 28 (Round 1)

```
Solution Response: Let the sequence be
    defined by $x_1 = \frac{25}{11}$ and
    $x_{k+1} = \frac{1}{3}\left(x_k +
    \frac{1}{x_k} - 1\right)$.
This simplifies to $x_{k+1} = \frac{x_k^2 -
    x_k + 1}{3x_k}$.
Let $x_k = \frac{m_k}{n_k}$ with $\gcd(m_k,
    n_k) = 1$. Then
\[ x_{k+1} = \frac{m_k^2 - m_k n_k +
    n_k^2}{3m_k n_k}. \]
Let $m_{k+1}' = m_k^2 - m_k n_k + n_k^2$
    and $n_{k+1}' = 3m_k n_k$.
The sum of numerator and denominator is
    $S_{k+1}' = m_{k+1}' + n_{k+1}' = m_k^2
    + 2m_k n_k + n_k^2 = (m_k + n_k)^2 =
    S_k^2$.
The actual terms $m_{k+1}, n_{k+1}$ are
    obtained by dividing by $g_k =
    \gcd(m_{k+1}', n_{k+1}')$.
We have $\gcd(m_k^2 - m_k n_k + n_k^2, m_k
    n_k) = 1$ since $\gcd(m_k, n_k) = 1$.
Thus $g_k = \gcd((m_k+n_k)^2, 3)$.
Since $S_1 = 25+11=36$, which is divisible
    by 3, we have $g_1 = 3$.
Also, $S_{k+1} = S_k^2 / g_k$. If $S_k$ is
    divisible by 3, then $S_k^2$ is
    divisible by 9, so $S_{k+1} = S_k^2/3$
    is divisible by 3.
Thus by induction, $g_k=3$ for all $k$, and
    $S_{k+1} = S_k^2 / 3$.
Let $T_k = S_k/3$. Then $3 T_{k+1} = (3
    T_k)^2 / 3 = 3 T_k^2 \implies T_{k+1} =
    T_k^2$.
We have $S_1 = 36$, so $T_1 = 12$.
```

```
Then $T_k = 12^{2^{k-1}}$, and $S_k = 3
    \cdot 12^{2^{k-1}}$.
We need $S_{2025} = 3 \cdot 12^{2^{2024}}
    \pmod{1000}$.
Modulo 8: $S_{2025} = 3 \cdot 4^{2^{2024}}
    \cdot 3^{2^{2024}} \equiv 0 \pmod 8$
    (for large exponent).
Modulo 125: $\phi(125) = 100$. The exponent
    is $E = 2^{2024}$.
$2^{2024} \equiv 16 \pmod{100}$.
So $S_{2025} \equiv 3 \cdot 12^{16}
    \pmod{125}$.
$12^{16} \equiv 84 \pmod{125}$.
$S_{2025} \equiv 3 \cdot 84 = 252 \equiv 2
    \pmod{125}$.
We solve the system $x \equiv 0 \pmod 8$
    and $x \equiv 2 \pmod{125}$.
$x = 125k + 2$. $125k + 2 \equiv 5k + 2
    \equiv 0 \pmod 8 \implies 5k \equiv -2
    \equiv 6 \implies k \equiv 6 \pmod 8$.
$x = 125(6) + 2 = 750 + 2 = 752$.

\boxed{752}
```

Reasoning Process 4. Correct Reasoning Process of AIME25 Problem 28 (Round 2)

```
Solution Response: To solve for $x_{2025}$,
    we first analyze the recurrence
    relation
    $x_{k+1}=\frac{1}{3}\left(x_k+\frac{1}{x_k}-
    = \frac{x_k^2 - x_k + 1}{3x_k}$.

Let's find the fixed points of the
    transformation $f(x) = \frac{x^2 - x +
    1}{3x}$.
Setting $f(x) = x$, we get $x^2 - x + 1 =
    3x^2$, which implies $2x^2 + x - 1 =
    0$. Factoring gives $(2x-1)(x+1) = 0$.
    The fixed points are $x = 1/2$ and $x =
    -1$.

This suggests using a linear fractional
    transformation that maps these fixed
    points to $0$ and $\infty$ to simplify
    the recurrence. Let $z_k = \frac{x_k -
    1/2}{x_k + 1} = \frac{2x_k - 1}{2(x_k +
    1)}$. To avoid the factor of 2 in the
    denominator, let's define $z_k =
    \frac{2x_k - 1}{x_k + 1}$.

Let's find the recurrence for $z_k$:
$$z_{k+1} = \frac{2x_{k+1} - 1}{x_{k+1} +
    1} = \frac{2\left(\frac{x_k^2 - x_k +
    1}{3x_k}\right) - 1}{\frac{x_k^2 - x_k
    + 1}{3x_k} + 1} = \frac{2(x_k^2 - x_k +
    1) - 3x_k}{x_k^2 - x_k + 1 + 3x_k} =
    \frac{2x_k^2 - 5x_k + 2}{x_k^2 + 2x_k +
    1}$$
$$z_{k+1} = \frac{(2x_k - 1)(x_k - 2)}{(x_k
    + 1)^2} = \left(\frac{2x_k - 1}{x_k +
    1}\right) \frac{x_k - 2}{x_k + 1}$$
We can express $\frac{x_k - 2}{x_k + 1}$ in
    terms of $z_k$. From $z_k = \frac{2x_k
    - 1}{x_k + 1} = \frac{2(x_k+1) -
    3}{x_k+1} = 2 - \frac{3}{x_k+1}$, we
    have $\frac{3}{x_k+1} = 2 - z_k$.
Then $\frac{x_k - 2}{x_k + 1} = \frac{x_k +
    1 - 3}{x_k + 1} = 1 - \frac{3}{x_k+1} =
    1 - (2 - z_k) = z_k - 1$.
Substituting this back, we get:
$$z_{k+1} = z_k (z_k - 1)$$

We calculate the first term $z_1$:
$$x_1 = \frac{25}{11} \implies z_1 =
    \frac{2(25/11) - 1}{25/11 + 1} =
    \frac{50/11 - 11/11}{25/11 + 11/11} =
    \frac{39}{36} = \frac{13}{12}$$
We can represent $z_k$ as a fraction
    $N_k/D_k$.
$z_1 = \frac{13}{12}$.
$z_2 = \frac{13}{12} \left(\frac{13}{12} -
    1\right) = \frac{13}{144}$.
$z_3 = \frac{13}{144} \left(\frac{13}{144}
    - 1\right) = \frac{13 \cdot
    (-131)}{144^2}$.
The denominator of $z_k$ follows the
    pattern $D_k = 12^{2^{k-1}}$.
The numerator $N_k$ and denominator $D_k$
    are relatively prime (as derived in the
    thought process).

We need $x_{2025} = \frac{m}{n}$. Inverting
    the transformation $z =
    \frac{2x-1}{x+1}$:
$$z(x+1) = 2x - 1 \implies zx + z = 2x - 1
    \implies 1 + z = x(2 - z) \implies x =
    \frac{1+z}{2-z}$$
Substituting $z_{2025} = \frac{N}{D}$:
$$x_{2025} = \frac{1 + N/D}{2 - N/D} =
    \frac{D + N}{2D - N}$$
Since $\gcd(N, D) = 1$ and $N \equiv 1
    \pmod 3$, $D \equiv 0 \pmod 3$, the
    fraction is irreducible.
Thus $m = D + N$ and $n = 2D - N$.
The sum $m+n = (D+N) + (2D-N) = 3D$.
We need to calculate $3D_{2025}
    \pmod{1000}$.
```

```
$D_{2025} = 12^{2^{2024}}$. Let $E =
    2^{2024}$. We need $3 \cdot 12^E
    \pmod{1000}$.

By Chinese Remainder Theorem on $1000 = 8
    \times 125$:
1.  **Mod 8**: $12 \equiv 4 \pmod 8$. $12^E
    = 4^E$. Since $E \ge 2$, $4^E$ is
    divisible by $16$, so $12^E \equiv 0
    \pmod 8$. Thus $3 \cdot 12^E \equiv 0
    \pmod 8$.
2.  **Mod 125**: We need $3 \cdot 12^E
    \pmod{125}$.
    Compute $E \pmod{\phi(125)}$.
    $\phi(125) = 100$.
    $E = 2^{2024}$.
    $E \equiv 0 \pmod 4$.
    $E \pmod{25}$: $\phi(25) = 20$. $2024 =
    20 \times 101 + 4$. $E \equiv 2^4 = 16
    \pmod{25}$.
    Using CRT for mod 100: $x \equiv 0
    \pmod 4$ and $x \equiv 16 \pmod{25}$
    implies $x \equiv 16 \pmod{100}$.
    So we need $12^{16} \pmod{125}$.
    $12^2 = 144 \equiv 19$.
    $12^4 \equiv 19^2 = 361 \equiv 111
    \equiv -14$.
    $12^8 \equiv (-14)^2 = 196 \equiv 71$.
    $12^{16} \equiv 71^2 = 5041 = 40 \times
    125 + 41 \equiv 41$.
    So $3 \cdot 12^E \equiv 3 \cdot 41 =
    123 \pmod{125}$.

Solving for $X \equiv 0 \pmod 8$ and $X
    \equiv 123 \pmod{125}$:
$X = 125k + 123$.
$125k + 123 \equiv 5k + 3 \equiv 0 \pmod 8
    \implies 5k \equiv -3 \equiv 5 \implies
    k \equiv 1 \pmod 8$.
$X = 125(1) + 123 = 248$.

The remainder is 248.

\boxed{248}
```

H.3 Example Reasoning Traces in Section 4.4

Reasoning Trace 1. Inverse Frequency Balancing

```
    This method flattens long-tailed
    (Zipfian) data distributions to prevent
    models from overfitting to common
    'head' concepts while underfitting rare
    'tail' concepts. It uses a
    probabilistic sampling approach based
    on a maximum frequency threshold to
    ensure a more uniform distribution of
    concepts. When to use: When training
    foundation models or classifiers on
    natural data distributions where a
    small number of classes or concepts
    dominate the dataset (e.g., millions of
    'cat' images vs. few 'specific
    species'), and the goal is to maximize
    zero-shot generalization or
    task-agnostic performance.
    Step-by-step: 1) Scan the dataset to
    calculate the total count ($N$) for
    every specific metadata concept or
    class 2) Define a saturation threshold
    ($t$) that represents the desired
    maximum number of samples per concept
    (e.g., $t=20,000$ was used for 400M
    total items) 3) Compute a sampling
    probability for each concept using the
    formula $P = \\min(1, t/N)$, ensuring
    tail concepts are kept with probability
    1 and head concepts are downsampled 4)
    Iterate through the data points again;
    for items associated with multiple
    concepts, generate a random number
    against the probability of each
    associated concept 5) Select the data
    point for the final training set if *at
    least one* of its associated concepts
    triggers a selection, thereby
    preserving diversity even within common
    items.
```

Reasoning Trace 2. Directional Skill Evolution

```
This technique involves systematically
    transforming specific, verified
    solutions into more general and
    reusable tools (skills) to enrich a
    knowledge base. Instead of just storing
    a specific answer, the system actively
    'evolves' it to cover broader cases.
    When to use: When building a knowledge
    base or skill library from specific
    examples, particularly in domains like
    mathematics or coding where a specific
    solution (e.g., for 2D vectors) often
    hints at a general principle (e.g., for
    N-dimensions). It is used to
    proactively expand the capabilities of
    an AI agent without requiring new
    external problem sets. Step-by-step: 1)
    Select a specific, verified skill or
    lemma from the existing library (the
    seed) 2) Choose one of four specific
    evolution heuristics: 'Parameterize'
    (replace specific constants like
    numbers with variables), 'Extend
    Dimensions' (generalize from specific
    dimensions to n-dimensions), 'Scale
    Complexity' (create a more complex
    version of the problem), or 'Identify
    Key Concepts' (extract and formalize
    the core underlying principle) 3)
    Retrieve relevant unsolved problems or
    requests to guide the evolution
    context, ensuring the new skill aims at
    actual gaps 4) Prompt the model to
    rewrite the seed skill according to the
    chosen heuristic 5) Formally verify the
    generated skill; if valid and distinct
    from existing skills, add it to the
    library.
```