COP: A Memory-Augmented Chain-of-Paradigms Framework for Inductive Reasoning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) demonstrate strong capabilities in text generation and reasoning, yet their reasoning often relies on prompt engineering or repeated model queries, limiting efficiency and cross-task generalization. Inspired by cognitive theories of Human Inductive Reasoning, we propose Chain of Paradigms (COP), a memory-augmented reasoning framework that enables LLMs to abstract and reuse high-level reasoning patterns across tasks. COP introduces a lightweight paradigm buffer to store structured reasoning patterns, a problem expander to extract task-critical information, and a semantic retrieval mechanism that dynamically selects relevant paradigms during reasoning. This design separates pattern abstraction from pattern application, enabling reusable reasoning structures without task-specific prompt design. Experiments on the Big-Bench Hard (BBH) benchmark demonstrate that COP consistently improves reasoning accuracy and robustness over prompting-based, multi-query, and tool-assisted baselines while maintaining efficient reasoning. Our results suggest that integrating Inductive Reasoning mechanisms can improve the reliability and generalization of LLM reasoning.

1 Introduction

Research Background and Significance. Generative models have shown strong capabilities in natural language processing (Bhatia, 2023), particularly in text generation, question-answering systems, and conversational agents. However, Large Language Models (LLMs) can predict logical relationships between sentences (e.g., entailment or contradiction) in natural language reasoning tasks (Sternberg, 1988), they fail to replicate Human Inductive Reasoning mechanisms observed in psychological studies (Fierro Celis and Andrade Navia, 2022). This highlights that the enhanced "planning abilit" of LLMs is essentially an optimization of approximate retrieval, rather than a true understanding of the problem's logic (García-Campos and Sarabia-López, 2022).

Theory and Motivational Foundations. Inductive Reasoning plays a central role in human cognition (Xu et al., 2025), allowing us to distill general laws from limited examples. Integrating this capability into LLMs not only enhances the models' reasoning ability but also better adapts them to complex tasks. Furthermore, by constructing reusable reasoning data patterns, we provide structured knowledge support for subsequent model generation tasks (Huang and Chang, 2023a), promoting the deeper development of LLMs in language understanding and generation. Moreover, this research offers new perspectives and methods for interdisciplinary fields such as brain science and cognitive psychology (Rajani et al., 2019).

Research Problems and Objectives. However, existing research has focused on improving the planning reasoning abilities of LLMs through prompt engineering and multi-query reasoning methods, but these approaches lack generalization and scalability (Choi et al., 2023; Chen et al., 2024; Kambhampati, 2024; Fodor and Pylyshyn, 1988). For example, single-query reasoning approaches (e.g., CoT (Wei et al., 2022) and few-shot cueing (Wang et al., 2020)) require task-specific cue design and lack generalizability. Multi-query reasoning methods (e.g., Least-to-Most (Zhou et al., 2023), Tree-of-Thoughts (Yao et al., 2023a), Graph-of-Thoughts (Besta et al., 2024), Question Rephrasing (Li et al., 2024)) solve complex problems by decomposing them into sub-problems but require multiple calls to the model (Huang and Chang, 2023b).

This study aims to design a new Chain of Paradigms (COP) method to enhance the Inductive Reasoning capabilities of LLMs by introducing an Inductive Reasoning paradigm. Our contributions include: (*i*) we incorporate Inductive

Reasoning into the generation process of generative models. (*ii*) we design a reasoning data structure based on the inductive paradigm. (*iii*) COP is driven by **memory-augmented inductive abstraction**, enabling automatic adaptation of reasoning paradigms across tasks without task-specific prompt engineering. (*iv*) We constructing reusable reasoning data maps and designing generative models for unknown reasoning paradigms.

2 Related Work

2.1 Multi-Query Reasoning.

Multi-query reasoning methods decompose complex problems by querying LLMs multiple times to generate various reasoning paths. Specifically, these methods include subproblem decomposition with iterative hinting (Dua et al., 2022), Tree of Thought (ToT) (Yao et al., 2023a), Graph-of-Thoughts (GoT) (Besta et al., 2024) and path search, retrieval enhancement and knowledge graph fusion, as well as self-consistency and diversity optimization (Wang et al., 2023).

For example, the iterative hinting technique allows subsequent subproblems to access antecedent results. ToT and GoT methods improve model reasoning capability by constructing a tree or graphical reasoning structure. The RoG framework (Luo et al., 2024), SearChain (Jiang et al., 2020), and Readi (Cheng et al., 2024) frameworks improve model reasoning by fusing query reasoning with knowledge graphs. However, these approaches are computationally expensive and rely on manually designed reasoning structures, which lack flexibility.

An alternative approach to complex reasoning delegates algorithmic computation to external tools, particularly code interpreters (Gao et al., 2023; Chen et al., 2022). Program-Aided Language Models (PAL) (Gao et al., 2023) generate Python programs to solve mathematical and symbolic reasoning tasks, and achieve high accuracy on algorithmic problems. However, tool-based methods face limitations: (1) they require executable specifications, excluding open-ended reasoning tasks; (2) they do not accumulate reusable knowledge across problems; and (3) they incur overhead from code execution.

COP complements tool-use by capturing reusable reasoning patterns that generalize across both algorithmic and non-algorithmic tasks, without requiring execution environments.

2.2 Inductive Reasoning Paradigm Interpretation.

Inductive Reasoning is a fundamental cognitive process that derives general principles or patterns from a finite set of observed instances (Misrom et al., 2020). Unlike deductive reasoning, which applies predefined rules to specific cases, Inductive Reasoning infers latent regularities from empirical evidence and extends them to unseen situations. This process has been widely adopted in qualitative research and cognitive science, where general theories are constructed through the analysis and abstraction of concrete examples (Peltonen, 2022; Mott and Bullock, 2015). By encouraging higher-order reasoning and pattern abstraction (Riguzzi et al., 2014; Hammer, 2011), Inductive Reasoning enhances problem-solving capabilities across diverse domains. For more details, see Appendix C.

2.3 Cognitive Foundations of Inductive Reasoning

Inductive Reasoning has been extensively studied in cognitive science as the process of inferring general principles from finite observations (Tenenbaum et al., 2006). Recent advances in program synthesis have demonstrated that humans and machines can induce reusable program abstractions from sparse examples (Ellis et al., 2020, 2023; Griffiths et al., 2010). The DreamCoder framework (Bayesian program learning) (Ellis et al., 2023) implements a wake-sleep algorithm: during wake, learned programs solve new tasks; during sleep, the system abstracts common patterns into a reusable library. LINC (Olausson et al., 2023; Garcez and Lamb, 2020) demonstrates that coupling language models with first-order logic provers improves logical reasoning while maintaining interpretability.

3 Method

As shown in Figure 1, COP is grounded in the cognitive science insight that human reasoning relies on reusable mental schemas abstracted from prior experience (Tenenbaum et al., 2006). Rather than reasoning from first principles for every problem, humans store and instantiate abstract patterns capacity known as *inductive generalization*. COP operationalizes this mechanism through a computational architecture that separates pattern discovery (sleep phase) from pattern application (wake

phase), drawing on Bayesian program learning (Ellis et al., 2023) and neuro-symbolic AI (Olausson et al., 2023). This cognitive grounding distinguishes COP from purely engineering-driven prompting methods.

3.1 Formal Framework and Notation

We introduce a unified data structure for modeling Inductive Reasoning processes. COP decomposes inductive reasoning into six interacting components:

$$COP = \{Q, A, Co, Ca, P, R\} \tag{1}$$

This formulation formalizes the inductive pipeline: structured observations are first extracted from the input question, generalized into an abstract pattern, and subsequently instantiated to generate the final answer.

$$\begin{aligned} \Phi(Q) &= (Co,\, Ca), \\ P &\sim \Pr(P \mid \Phi(Q)), \\ A &= \Psi(Q, P), \\ R &= \mathbb{I}\left[\mathrm{Sim}(P, \Phi(Q)) \geq \tau\right]. \end{aligned} \tag{2}$$

Here, $\Phi(\cdot)$ denotes an **abstraction operator** that converts an input question into structured reasoning traces (Co) and concrete cases (Ca). $\Psi(\cdot,\cdot)$ denotes a **generation operator** that instantiates pattern P with question-specific details to produce the final answer. $\mathrm{Sim}(\cdot,\cdot)$ measures semantic consistency using embedding-based cosine similarity, and τ is a threshold controlling the minimal consistency required for pattern validity.

Components. Q denotes the input problem and A the generated answer. Co represents reasoning traces (Chain-of-Thought), while Ca denotes concrete cases extracted from the question. P is the abstract reasoning pattern induced from (Co, Ca). R indicates whether the induced pattern remains consistent under similarity threshold τ. $R = 1$ indicates the pattern satisfies the similarity threshold τ.

From a formal perspective, COP models Inductive Reasoning as a conditional generalization process. Given observations composed of reasoning traces and cases (Co, Ca), the objective is to induce a reusable pattern P that maximizes explanatory consistency:

$$P^* = \arg\max_{P \in \mathcal{P}} \Pr(P \mid \Phi(Q)), \tag{3}$$

This formulation explicitly connects COP to probabilistic inductive reasoning, where patterns are treated as hypotheses conditional on observed evidence.

3.2 Operational Pipeline

COP operates in two phases that separate pattern discovery from pattern reuse.

Build Phase (Sleep). Training tasks are processed offline to populate the paradigm buffer. The abstraction operator Φ extracts (Co, Ca) and the paradigm manager induces reusable patterns. This phase corresponds to the "sleep" phase in DreamCoder (Ellis et al., 2023), where the system compresses experience into reusable abstractions.

reasoning Phase (Wake). Given a test question, relevant paradigms are retrieved from the buffer and instantiated by Ψ to generate the final answer. This separation ensures that the paradigm buffer (PB) serves as a pre-computed inductive memory, enabling efficient few-shot reasoning without requiring the LLM to rediscover patterns from scratch for each new task.

3.3 Question Expander

The question expander implements the abstraction operator Φ, serving as the initial inductive step of COP. It extracts structured observations from the input task and approximates inductive pattern reasoning. Specifically, it transforms raw questions into candidate paradigms by decomposing them into reasoning traces and cases.

Problem expansion. The expander decomposes Q into a reasoning trace Co and a set of cases Ca using an LLM with a structured extraction prompt ϕ:

$$(Co,\, Ca) = \Phi(Q) \approx \mathrm{LLM}_{\phi}(Q). \tag{4}$$

Pattern abstraction. The extracted evidence (Co, Ca) and **verification signals** $v \in \{0, 1\}$ indicates whether the reasoning trace produces the correct answer ($v = 1$ if $\mathrm{Execute}(Co) = A_{gt}$). These signals help filter spurious patterns during abstraction.

Key information extraction. From the expanded results, we extract implicit information $E = \{Co, Ca, v\}$, where $v \in \{0, 1\}$ denotes verification signals indicating whether the reasoning trace Co correctly leads to the true answer. We then form a task descriptor $\hat{T} = (E, p)$, where p represents paradigm features. This descriptor is abstracted into a meta-pattern capable of generalizing across structurally similar tasks.

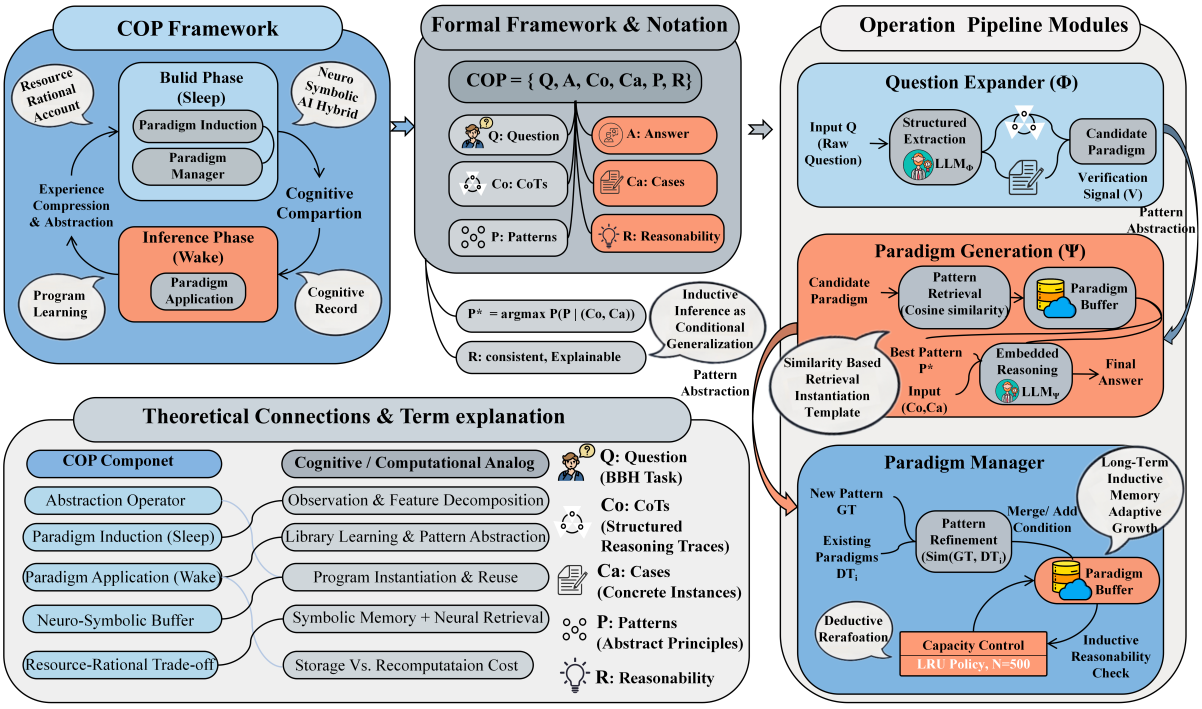

Figure 1: we first formalize the framework's mathematical structure (Section 3.1), and finally detail its operational modules: question expansion (Section 3.3), paradigm generation (Section 3.4), and paradigm management (Section 3.5).

3.4 Paradigm Generator

The paradigm generator implements the retrieval and instantiation mechanisms that enable pattern reuse across tasks.

Pattern retrieval. Given a candidate paradigm $\hat{T}$ (represented as an embedding $f(\hat{T})$), relevant paradigms are retrieved from the buffer via embedding similarity. We define the similarity function $\mathrm{Sim}(\cdot,\cdot)$ as cosine similarity between embeddings:

$$\mathrm{Sim}(f(\hat{T}), f(DT_i)) = \frac{f(\hat{T}) \cdot f(DT_i)}{\|f(\hat{T})\| \|f(DT_i)\|}. \quad (5)$$

The set of candidate paradigms is defined as:

$$GoodPs = \{DT_i \in PB \mid \mathrm{Sim}(f(\hat{T}), f(DT_i)) \geq \tau\}, \quad (6)$$

where PB denotes the paradigm buffer and $f(\cdot)$ is a sentence embedding model (e.g., MPNet).
The selected pattern is:

$$P^* = \arg \max_{P \in GoodPs} \Pr(P \mid \Psi(Q)). \quad (7)$$

Embedded reasoning. The selected pattern P^* is instantiated by the generation operator Ψ, which injects task-specific values from (Co, Ca) into pattern placeholders, yielding a concrete reasoning path that guides final answer generation.

$$A = \Psi(Q, P^*) \approx \mathrm{LLM}_\psi(Q, P^*), \quad (8)$$

where ψ denotes the instantiation prompt template. The prompt construction template based on Inductive Reasoning mapping is provided in Appendix K.2.

3.5 Paradigm Manager

We define a *paradigm* as a high-level inductive reasoning template distilled from observed tasks. Formally, each paradigm is represented as a tuple:

$$DT = (P, m), \quad (9)$$

where m encodes metadata such as creation timestamp, usage count, and originating task. The set of all paradigms constitutes the paradigm buffer: $PB = \{DT_1, DT_2, \ldots, DT_N\}$. Each paradigm $DT_i = (A_i, m_i)$ encodes an abstraction A_i with metadata m_i (creation time, usage count, task source), serving as long-term inductive memory that supports cross-task generalization. During updates, we identify the most compatible existing paradigm $GDT \in PB$ for potential merging (Eq. (12)).

The manager maintains buffer quality through two principles: **Pattern refinement** and **capacity control**.

Pattern refinement. Given a newly induced pattern GT, its compatibility with existing paradigms is evaluated:

$$\gamma_i = \mathrm{Sim}(GT, DT_i) = \frac{f(GT) \cdot f(DT_i)}{\|f(GT)\| \|f(DT_i)\|} \tag{10}$$

Let $\gamma^* = \max_i \gamma_i$. If $\gamma^* < \tau$, GT is considered novel and admitted to PB:

$$PB \leftarrow PB \cup \{GT\}. \tag{11}$$

Otherwise, the most compatible paradigm is selected for potential merging:

$$GDT = \arg \max_{DT_i \in PB} \gamma_i. \tag{12}$$

This update rule ensures that PB maintains a diverse yet compact set of paradigms: each $DT_i \in PB$ represents a distinct inductive pattern, and the buffer grows only when novel reasoning structures are encountered.

Capacity control. To balance generalization and efficiency, a Least Recently Used (LRU) policy maintains the buffer within fixed capacity N. In all experiments, $N = 500$. Limited capacity biases the system toward recent patterns, favoring short-term adaptation; increasing N preserves low-frequency but high-transfer paradigms, enabling cross-task generalization.

4 Experiments and Evaluation

4.1 Experimental Setup

Experimental Objectives. We investigate three complementary aspects of COP:[1](RQ1) whether paradigm-level inductive abstraction improves reasoning accuracy across heterogeneous tasks; (RQ2) whether COP enables effective cross-task transfer of reusable reasoning paradigms; and (RQ3) whether paradigm reuse improves reasoning efficiency compared to multi-step prompting or agent-style strategies.

Dataset. We evaluate on BIG-Bench Hard (Srivastava et al., 2023), a challenging subset of 23 tasks selected for (i) <80% accuracy by existing LLMs at the time of dataset creation, (ii) diversity in required reasoning types (deductive, inductive, abductive), and (iii) solvability by human experts without specialized domain knowledge.

Baselines. We compare against four categories of methods. *Prompting-based methods* include Zero-shot, Chain-of-Thought (CoT), Question Rephrasing (QReph), Reverse Reasoning, ReAct (Yao et al., 2023b), Reflexion, Tree-of-Thoughts (ToT), and Graph-of-Thoughts (GoT). *Retrieval-based methods* implement dense retrieval using jina-embeddings-v2 without explicit inductive abstraction. *Tool-use methods* comprise Program-Aided Language models (PAL-Code and PAL-Math) (Gao et al., 2023), where LLMs generate executable Python programs. *Model scaling* experiments span Yi-34B-Chat, Llama-3.x (1B/3B/8B), and Qwen variants (Plus/Turbo).

Implementation Details. All experiments use the HuggingFace Transformers framework with jina-embeddings-v2 as the default encoder (768-dimensional paradigm representations). The paradigm buffer updates dynamically every 100 tasks via incremental memory replacement of low-frequency paradigms. Training data mixes within-task and cross-task examples at a 7:3 ratio to balance specialization and generalization. All experiments were run on two NVIDIA GTX 1080Ti GPUs (12GB); hyperparameters and prompts are detailed in Appendix H.

Evaluation Metrics. We report Exact Match (EM) for task accuracy, Average Reasoning Time (ART) for end-to-end latency (including embedding computation, retrieval, and reasoning), and Cross-task Template Reuse Rate (CTRR) to quantify inductive transfer. CTRR measures the proportion of queries where the retrieved paradigm originates from a different task: CTRR $= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[T(q_i) \neq T(P(q_i))]$, where $T(\cdot)$ denotes task label and $P(q_i)$ the selected paradigm for query q_i.

Build-reasoning Separation. To isolate the contribution of paradigm quality from base model capability, we adopt a strict two-phase design. *Build Phase:* Paradigm extraction uses Qwen-Plus offline to populate the buffer; this occurs once and is independent of test-time models. *reasoning Phase:* Reported LLMs (Llama-1B/3B/8B, Qwen-Plus/Turbo) perform only pattern instantiation and answer generation, enabling fair comparison across model scales.

[1]Detailed implementation details and simulation examples are provided in the supplementary material and link

Method	Llama-1B		Llama-3B		Llama-8B		Yi-34B		Qwen-Plus		Qwen-Turbo	
	EM	ART	EM	ART	EM	ART	EM	ART	EM	ART	EM	ART
Zero-shot	32.3	**18.0**	26.1	**19.0**	47.6	**21.0**	21.1	**25.0**	35.7	**28.0**	22.1	**26.0**
CoT	35.7	30.2	30.0	32.2	40.5	35.1	24.5	38.2	58.2	40.3	28.2	39.3
QReph	44.5	45.3	35.2	47.3	40.5	50.4	29.9	55.3	64.9	58.4	52.2	56.3
Reverse	39.5	55.4	39.5	58.3	47.6	62.5	34.8	65.4	74.4	68.5	61.3	66.4
ReAct	41.2	60.5	38.4	62.4	46.1	65.6	36.8	68.4	72.3	70.5	63.4	69.3
Reflexion	42.1	65.6	39.3	68.5	46.8	72.7	38.1	75.6	73.6	78.5	64.9	76.4
ToT	43.8	78.7	40.9	82.6	47.3	88.7	41.6	92.6	74.2	95.8	65.7	93.7
GoT	44.2	85.8	41.1	88.7	47.8	92.8	42.3	96.7	74.9	98.9	66.1	97.9
PAL-Code	48.5	25.3	52.1	28.4	61.3	32.5	55.2	35.1	76.8	42.3	68.4	40.1
PAL-Math	50.2	24.1	54.3	26.8	63.5	30.2	57.1	33.4	78.2	40.5	70.1	38.7
Dense Retrieval	40.5	52.1	42.0	54.3	48.0	58.6	38.2	61.4	59.7	62.3	50.8	60.1
COP (FC)	**52.6**	105.8	41.3	93.4	48.3	96.5	45.8	108.7	**75.4**	111.8	**66.0**	109.3
COP (DLRU)	45.8	98.4	41.0	101.3	48.1	104.5	45.3	112.4	72.1	115.6	62.8	113.2
COP (MPO)	46.5	85.4	**41.5**	88.5	**48.5**	92.4	**46.0**	105.6	73.7	108.6	63.3	106.5

Table 1: SOTA Comparison of Reasoning Methods on Complex Reasoning Benchmarks. EM = Exact Match (%), ART = Average Reasoning Time (s). Blue cells indicate best EM per model, gray cells indicate low EM baselines. **Implementation Note:** For all COP variants, paradigm extraction (Build Phase) is performed using Qwen-Plus to ensure high-quality pattern abstraction. The reported LLMs (Llama-1B/3B/8B, etc.) are used only in the reasoning Phase for pattern instantiation and answer generation. This explains the counter-intuitive result where Llama-1B achieves higher accuracy than Llama-3B, as both rely on the same pre-computed paradigm buffer. Clarification: **PAL-Math and PAL-Code results are reported only for BBH mathematical tasks**.

4.2 Evaluation on Complex Reasoning

Setup. We compare COP with representative prompting, agent-based, tool-assisted, and retrieval-based reasoning methods across six language models ranging from 1B parameters to API-scale models.

Main Results. Table 1 presents exact match (EM) accuracy and average reasoning time (ART). Three key findings emerge:

First, paradigm reuse substantially improves smaller models. COP achieves the highest EM scores on Llama-1B (52.6%) and Llama-3B (41.5%), outperforming the strongest Prompting-based methods baseline by 8.1% and 2.0%, respectively. This demonstrates that structured inductive abstraction effectively compensates for limited parametric capacity.

Second, tool-use methods challenge COP on algorithmic tasks but lack generality. PAL-Code and PAL-Math achieve strong accuracy on mid-scale models (Llama-3B: 54.3% EM; Llama-8B: 63.5%) with one-third the latency than COP (28s vs. 93s ART). However, their reliance on executable specifications excludes semantic reasoning tasks (e.g., Causal Judgment: 31.4% vs. 67.8% for COP). This reveals complementary strengths: tool-use excels in structured domains, while COP provides broader coverage across both algorithmic and open-ended Inductive Reasoning.

Third, COP provides a favorable accuracyefficiency trade-off among general-purpose methods. As shown in Figure 2 (a) and (b), PAL-Math achieves the lowest latency but requires executable specifications, limiting its applicability. Reverse prompting attains the highest accuracy but relies on task-specific prompt engineering. outperforming agent-based methods in speed while covering a broader range of tasks than tool-dependent approaches like PAL.

Fourth, gains diminish on instruction-tuned API models. On Qwen-Plus and Qwen-Turbo, COP provides only marginal improvements over strong prompting baselines. Large instruction-tuned models may already internalize reusable reasoning patterns, reducing the marginal benefit of explicit paradigm reuse.

Model Scaling Analysis. Table 1 reveals an unexpected pattern: Llama-1B achieves higher EM (52.6%) than Llama-3B (41.3%) under COP (FC). Since paradigms are extracted offline using Qwen-Plus, model differences reflect only pattern instantiation ability. Smaller models appear more receptive to external guidance, while larger models rely more heavily on parametric knowledge.

reasoning Cost Structure. COP exhibits stable reasoning time across model scales (105.8s for Llama-1B vs. 111.8s for Qwen-Plus). This is dominated by a fixed retrieval overhead ($\approx$ 80–

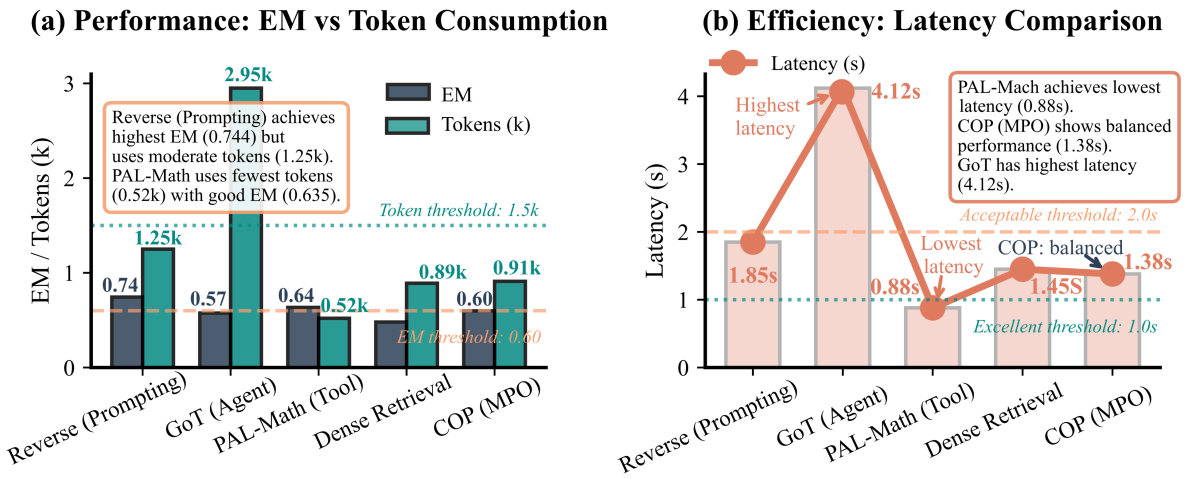

Figure 2: Accuracy-efficiency trade-off across method categories (aggregated across Llama-1B/3B/8B, Yi-34B, Table compares the strongest method from each category). EM = Exact Match accuracy averaged across models; Tokens = average tokens per query; Latency = wall-clock time in seconds. Best results per metric in **bold**.

90 s) for embedding computation and similarity search, making COP more suitable for batch reasoning rather than latency-sensitive applications.

4.3 Evaluation on Cross-Task Generalization

Impact of Buffer Capacity. Table 2 examines how paradigm buffer capacity ($N \in \{50, 200, 500\}$) and management strategies affect generalization. Increasing capacity consistently improves cross-task reuse rates (CTRR) across all strategies, confirming that richer paradigm repositories enhance transfer. However, larger buffers increase retrieval latency and memory consumption, revealing a coverage-efficiency trade-off.

Comparison of Management Strategies. Mixed Phase-out achieves the best accuracy efficiency trade-off, while Dynamic LRU minimizes latency but sacrifices cross-task transfer. Fixed Capacity preserves paradigm diversity but adapts less effectively under resource constraints.

Ablation on Buffer Design. Dynamic LRU degrades at large capacity ($N = 500$), likely due to retrieval noise combined with recency bias that evicts low-frequency but transferable paradigms. In contrast, Fixed Capacity benefits consistently from larger buffers.

5 Ablation Study

Module Contributions. Figure 3 examines each COP component through isolation experiments (positive contribution) and removal tests (negative ablation).

Positive contributions. As shown in Figure 3 (a)(b), the Question Expander improves EM by structuring reasoning traces but provides no cross-task transfer. Retrieval alone enables moderate transfer but suffers from rigid pattern matching. Instantiation with random paradigms improves reasoning structure but fails without relevant retrieval.

Negative ablation. Removing the paradigm buffer causes the largest performance drop (-8.5% EM), confirming that structured memory is essential for inductive transfer. Removing template selection further degrades accuracy, while removing capacity control reduces latency but harms performance due to buffer saturation.

Synergy Analysis. The full system, achieving 52.6% EM, substantially outperforms any individual module or the average of individual modules ($\approx$39.8%), demonstrating that COP gains emerge from component interaction: Φ enables effective similarity matching, retrieval identifies relevant patterns, and Ψ adapts them to new contexts.

Efficiency Trade-offs. As illustrated in Figure 3 (c)(d), the full COP system achieves the highest EM and CTRR at the cost of 14.3% more tokens (120k vs. 105k) and 19.8% slower reasoning (4.97s vs. 4.15s). Notably, removing capacity control reduces latency to near-baseline (4.20s) with marginal performance loss, offering a practical efficiency optimization for deployment.

6 Failure Analysis

Although COP improves reasoning performance across benchmarks, our analysis of 500 failure

Strategy	Capacity N	EM (%)	ART (s)	CTRR (%)	Retrieval Time (s)	Memory (MB)
COP (Fixed Capacity)	50	**38.67**	**86.3**	24.8	2.34	230
	200	46.2	90.4	58.8	3.69	460
	500	**52.6**	**105.9**	**68.4**	**4.97**	690
COP (Dynamic LRU)	50	**44.3**	**81.2**	21.6	2.27	220
	200	**48.7**	82.5	60.4	3.38	420
	500	46.5	**85.4**	66.3	4.68	650
COP (Mixed Phase-out)	50	38.7	**87.2**	**19.4**	**2.19**	200
	200	42.3	94.8	58.7	3.26	350
	500	**45.8**	**98.4**	65.1	4.53	600

Table 2: Cross-task Generalization Results: Impact of Paradigm Buffer Capacity and Management Strategy. EM = Exact Match (%), ART = Average Reasoning Time (s), CTRR = Cross-task Reuse Rate (%).

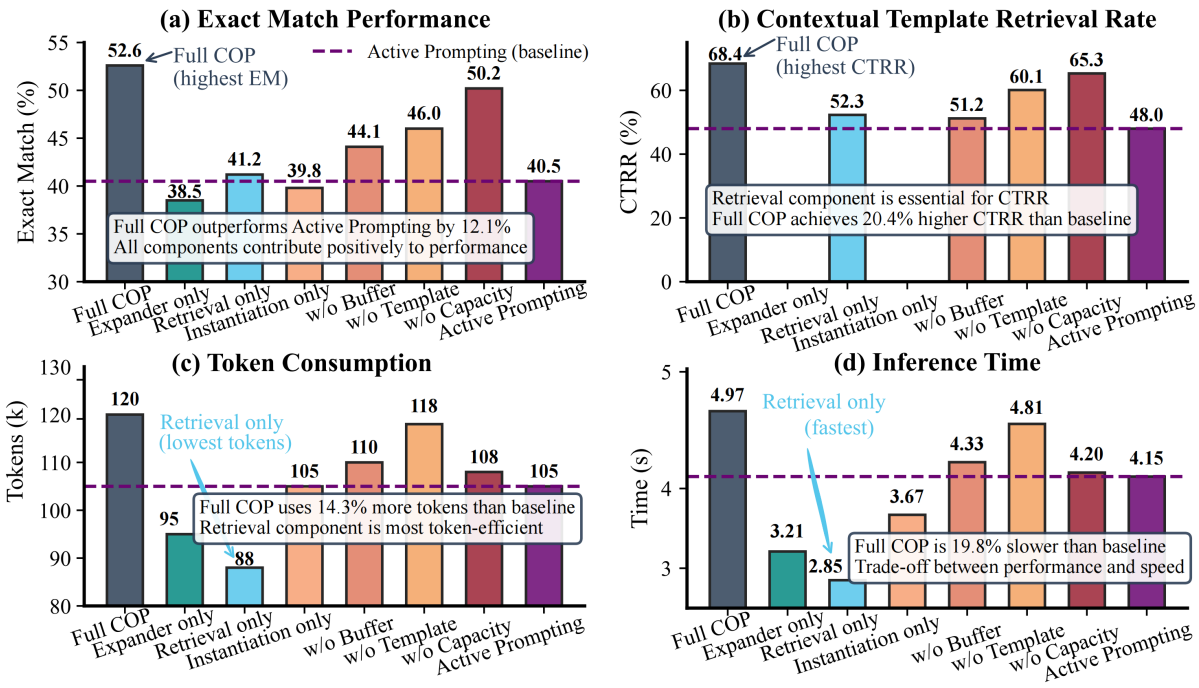

Figure 3: Full COP achieves best performance (52.6% EM, 68.4% CTRR) but at higher computational cost. All components contribute meaningfully, with retrieval being essential for CTRR. Ablation reveals the importance of each component in the overall framework.

cases reveals several systematic limitations. First, paradigm retrieval may select semantically similar but procedurally incompatible patterns, leading to incorrect reasoning trajectories. Second, the abstraction operator can occasionally generate overly generic paradigms that lack actionable structure, reducing transfer effectiveness. Third, large paradigm buffers introduce retrieval competition among similar patterns, causing unstable selection. Finally, the benefit of explicit paradigms diminishes for highly capable instruction-tuned models that may already internalize similar reasoning patterns. Detailed examples, quantitative distributions, and mitigation strategies are provided in Appendix D.

7 Conclusion

We proposed Chain of Paradigms (COP), a structured reasoning framework that enhances inductive transfer in LLMs by storing and dynamically retrieving reusable high-level paradigms. Experiments on BIG-Bench Hard showed that COP consistently improves reasoning accuracy in small-to-medium models while maintaining reasoning efficiency. Paradigm memory and buffer management are critical to cross-task generalization. These findings establish explicit paradigm abstraction as a promising direction for improving inductive reasoning in large language models, and open up avenues on integrating structured memory with reasoning-optimized architectures.

8 Limitations

Despite the promising results of the Chain-of-Paradigms (COP) framework, several limitations remain. We analyze these along three axes: the framework's generalization to other reasoning types, its applicability to resource-constrained deployments, and its robustness under continual learning settings.

Generalization to Other Reasoning Types. While COP is designed for Inductive Reasoning, its paradigm reuse mechanism could theoretically extend to deductive and abductive tasks. However, deductive reasoning requires soundness guarantees that COP's similarity-based retrieval cannot provide, suggesting integration with formal verifiers (e.g., LINC (Olausson et al., 2023)) would be necessary. For abductive reasoning, COP could store explanation patterns, but the non-monotonic nature of abduction introduces challenges for paradigm conflict resolution. Furthermore, real-time conversational deployment remains infeasible due to latency from online paradigm extraction. A hybrid approach pre computing domain paradigms offline and retrieving contextually may balance adaptability and responsiveness, but this remains future work. Consequently, COP is currently limited to offline batch processing of tasks with stable underlying patterns.

Integration with Reasoning-Optimized Models. Recent models like OpenAI o1 and DeepSeek-R1 achieve strong performance through extensive test-time computation and reinforcement learning on chain-of-thought trajectories. While direct comparison with closed-source o1 is infeasible, preliminary experiments show that COP provides minimal gains (<2% exact match) when applied to DeepSeek-R1-Distill models. This suggests that distilled models may already internalize effective reasoning patterns, reducing the headroom for paradigm-based improvement. However, the benefits of COP and test-time computation are potentially orthogonal: paradigms could guide the search process of reasoning models, reducing exploration space during generation. Future work should explore *adaptive paradigm application* selectively invoking paradigm guidance only when the model's confidence is low to combine COP's efficiency with the capability of reasoning models.

Robustness in Continual Learning Settings. Through controlled case studies, we identify three persistent challenges under memory-augmented continual learning. First, *paradigm drift* occurs when sequentially trained models overwrite previously acquired paradigms (e.g., numerical heuristics interfere with lexical sorting rules), indicating insufficient protection of task-specific inductive rules. Second, *incomplete rule transfer* emerges when paradigms lack granularity for depth-sensitive reasoning (e.g., lexical sorting of long words terminates prematurely), suggesting that the continual paradigm update mechanism overemphasizes high-frequency rules at the expense of fine-grained strategies. Third, *temporal inconsistency* arises when rules not frequently activated degrade over time (e.g., duplicate-handling rules applied inconsistently across learning stages), revealing that temporal alignment modules prioritize recent paradigms without enforcing long-term consistency.

Summary. These limitations point to three future directions: (i) strengthening paradigm isolation mechanisms to mitigate interference, (ii) incorporating depth-aware inductive constraints for fine-grained rule transfer, and (iii) introducing consistency-regularized paradigm replay strategies to maintain temporal coherence. Addressing these challenges may further bridge the gap between cognitive models of reasoning and modern large language model architectures.

9 Acknowledgments

We are grateful to the anonymous reviewers for their constructive comments and insightful suggestions, which substantially improved the clarity and rigor of this work. We are also grateful to colleagues and collaborators for valuable discussions on Inductive Reasoning, cognitive modeling, and the evaluation of Large Language Models. This research benefited from open-source tools and benchmark resources, including the BIG-Bench Hard (BBH) dataset, which enabled systematic evaluation and reproducible comparisons. Any remaining errors are the sole responsibility of the authors.

10 Ethical Considerations

All experiments in this study use the publicly available BBH benchmark. API calls to Baidu's Ernie Bot were conducted in accordance with its Terms of Service, and no personal or sensitive data were involved. This study did not require collection of human subjects data.

References

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

Sudeep Bhatia. 2023. Inductive reasoning in minds and machines. *Psychological Review*, pages 1373–1391.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Wenhu Chen, Xueguang Ma, Xinyun Liu, Yujia Lu, and William Yang Wang. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Zheyi Chen, Liuchang Xu, Hongting Zheng, Luyao Chen, Amr Tolba, Liang Zhao, Keping Yu, and Hailin Feng. 2024. Evolution and prospects of foundation models: From Large Language Models to large multimodal models. *Computers, Materials & Continua*, 80(2).

Sitao Cheng, Ziyuan Zhuang, Yong Xu, Fangkai Yang, Chaoyun Zhang, Xiaoting Qin, Xiang Huang, Ling Chen, Qingwei Lin, Dongmei Zhang, and 1 others. 2024. Call me when necessary: Llms can efficiently and faithfully reason over structured environments. *arXiv preprint arXiv:2403.08593*.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. 2023. Kcts: knowledge-constrained tree search decoding with token-level hallucination detection. *arXiv preprint arXiv:2310.09044*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. 2020. DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *arXiv preprint arXiv:2006.08381*.

Kevin Ellis, Lucas Wong, Maxwell Nye, Mathias Sablé-Meyer, Luc Cary, Lorena Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. 2023. DreamCoder: Growing generalizable, interpretable knowledge with wake–sleep Bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251):20220050.

Fernando Adolfo Fierro Celis and Juan Manuel Andrade Navia. 2022. Análisis de los prejuicios y errores en la toma de decisiones. un caso práctico en empresas de servicio. *Ciencias Administrativas, Econǿmicas y Contables*.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Artur S. d'Avila Garcez and Luís C. Lamb. 2020. Neurosymbolic ai: The 3rd wave. *arXiv preprint arXiv:2012.05876*.

Jonatan García-Campos and Saúl Sarabia-López. 2022. Tres grandes enigmas de los sesgos cognitivos. *SCIO: Revista De Filosofía*, (22):99–125.

Thomas L. Griffiths, Nick Chater, Charles Kemp, Amy Perfors, and Joshua B. Tenenbaum. 2010. Probabilistic models of cognition: exploring representations and inductive biases. *Trends in Cognitive Sciences*, 14(8):357–364.

Thomas L Griffiths, Falk Lieder, and Noah D Goodman. 2015. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 7(2):217–229.

Roger Julius Hammer. 2011. STRATEGY DEVELOPMENT PROCESS AND COMPLEX ADAPTIVE SYSTEMS.

Jie Huang and Kevin Chen-Chuan Chang. 2023a. Towards reasoning in Large Language Models: A survey. *Preprint*, arXiv:2212.10403.

Jie Huang and Kevin Chen-Chuan Chang. 2023b. Towards reasoning in large language models: A survey.

Peng Jiang, Fuchun Guo, Kaitai Liang, Jianchang Lai, and Qiaoyan Wen. 2020. Searchain: Blockchain-based private keyword search in decentralized

storage. *Future Generation Computer Systems*, 107:781–792.

Subbarao Kambhampati. 2024. Can Large Language Models reason and plan. *Annals of the New York Academy of Sciences*, 1534(1):15–18.

Emil O. W. Kirkegaard. 2009. Induction and a probability formula — emilkirkegaard.dk. https://emilkirkegaard.dk/en/2009/06/induction-and-a-probability-formula/. [Accessed 01-08-2025].

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024. Common 7b language models already possess strong math capabilities. *Preprint*, arXiv:2403.04706.

Falk Lieder and Thomas L Griffiths. 2020. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*.

Noor Suhaily Binti Misrom, Abdurrahman Sani Muhammad, Abdul Abdullah, Sharifah Osman, Mohd Hamzah, and Ahmad Fauzan. 2020. Enhancing students' higher-order thinking skills (hots) through an inductive reasoning strategy using geogebra. *International Journal of Emerging Technologies in Learning (iJET)*, 15(3):156–168.

John H. Mott and Darcy M. Bullock. 2015. Recommendations for improvement of collegiate flight training operational efficiency through guided-inquiry inductive learning. *International Journal of Aviation, Aeronautics, and Aerospace*, 2(4).

Theo X Olausson, Alex Gu, Benjamin Lipkin, Cedegao E Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176. Association for Computational Linguistics.

Tuomo Peltonen. 2022. Poppers critical rationalism as a response to the problem of induction: Predictive reasoning in the early stages of the covid-19 epidemic. 22(1):7–23.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*.

Fabrizio Riguzzi, Elena Bellodi, and Riccardo Zese. 2014. A history of probabilistic inductive logic programming. *Frontiers in Robotics and AI*, Volume 1 - 2014.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Robert J Sternberg. 1988. The psychology of human thought. *New York*.

Joshua B Tenenbaum, Thomas L Griffiths, and Charles Kemp. 2006. Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, 10(7):309–318.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-Thought prompting elicits reasoning in Large Language Models. *Advances in neural information processing systems*, 35:24824–24837.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, and 1 others. 2025. Towards large reasoning models: A survey of reinforced reasoning with Large Language Models. *arXiv preprint arXiv:2501.09686*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi.

2023. Least-to-most prompting enables complex reasoning in Large Language Models. *Preprint*, arXiv:2205.10625.

A COP Framework Description

Algorithm 1 Chain-of-Paradigm (COP) Inductive Reasoning Framework

Require: Input question Q, paradigm buffer $PB = \{DT_i\}_{i=1}^{N}$, buffer capacity K, reasonability threshold τ

Ensure: Final answer A, updated paradigm buffer PB

// Question Expander
Decompose Q into reasoning trace Co and cases Ca
Induce candidate paradigm $\hat{T} \leftarrow \text{LLM}(\psi(Co, Ca))$
// Pattern Retrieval
for all $DT_i \in PB$ **do**
  Compute similarity score $\gamma_i \leftarrow \text{Sim}(f(\hat{T}), f(DT_i))$
end for
$GoodPs \leftarrow \{DT_i \mid \gamma_i \text{ is top-ranked}\}$
// Best Pattern Selection
$P^* \leftarrow \arg\max_{P \in GoodPs} \Pr(P \mid Co, Ca)$
// Embedded Reasoning
Instantiate reasoning path using P^* and (Co, Ca)
Generate final answer A
// Paradigm Manager
$\gamma^* \leftarrow \max_i \gamma_i$
if $\gamma^* < \tau$ **then**
  Insert $\hat{T}$ into PB
else
  Merge $\hat{T}$ into $GDT = \arg\max_{DT_i \in PB} \gamma_i$
end if
if $|PB| > K$ **then**
  Remove least recently used paradigm from PB
end if
return A, PB

Algorithm 1 formalizes the end-to-end Inductive Reasoning and paradigm management process of the proposed Chain-of-Paradigm (COP) framework.

At reasoning time, COP first invokes the question expander to transform the raw input question Q into structured inductive observations, consisting of a reasoning trace Co and a set of concrete cases Ca. These elements serve as empirical evidence for approximating inductive reasoning, from which a candidate paradigm $\hat{T}$ is induced using a constrained LLM prompt.

Next, COP performs Pattern Retrieval by computing embedding-based similarity scores between $\hat{T}$ and historical paradigms stored in the paradigm buffer PB. This retrieval step serves as a practical approximation of posterior consistency in inductive reasoning, identifying paradigms that are structurally compatible with the current task.

Among the retrieved candidates, Best Pattern Selection (Line 10) chooses the paradigm P^* that maximizes conditional explanatory power given the current observations (Co, Ca). This step ensures that the selected paradigm is not only similar but also contextually appropriate.

The selected paradigm is then instantiated through Embedded Reasoning, where abstract pattern placeholders are filled with task-specific values, yielding a concrete reasoning trajectory that guides final answer generation.

Finally, the Paradigm Manager updates the paradigm buffer to support memory-augmented inductive learning. If the induced paradigm $\hat{T}$ is insufficiently explained by existing paradigms (i.e., $\gamma^* < \tau$), it is admitted as a new inductive unit. Otherwise, it is merged into the most compatible existing paradigm. To maintain efficiency and prevent unbounded growth, a Least Recently Used (LRU) policy enforces a fixed buffer capacity K.

Overall, this algorithm operationalizes Inductive Reasoning as a structured, reusable, and continually evolving process, bridging cognitive inductive theory with practical generative modeling.

B Module Interaction and Theoretical Connections

The Abstraction Operator Φ performs observation extraction and feature decomposition, drawing on theoretical foundations from theory-based inductive learning (Tenenbaum et al., 2006). During the sleep phase, Paradigm Induction enables library learning and pattern abstraction through a process analogous to the DreamCoder framework's abstraction mechanism (Ellis et al., 2023, 2020). The wake phase implements Paradigm Application, where learned patterns are instantiated and reused for efficient program synthesis. A Neuro-Symbolic Buffer serves as the system's memory architecture, combining symbolic memory storage with neural retrieval mechanisms (Olausson et al., 2023). The overall system operates according to Resource-Rational Trade-off principles, balancing storage costs against recomputation costs (Lieder and Griffiths, 2020). This integrated architecture forms a resource-rational system where the build phase amortizes pattern discovery costs during sleep, while the reasoning phase enables efficient reuse during wake, explaining why pattern storage outperforms de novo reasoning by exploiting recurring patterns across Inductive Reasoning tasks (Griffiths et al., 2015).

C Inductive Reasoning Paradigm Derivation

Formally, the inductive derivation process can be described as follows. Given a finite set of observations $S_1, S_2, \ldots, S_n \in S$, if each observation S_j exhibits a shared property or pattern P_{pat}, i.e.,

$$S_j \to P_{\text{pat}}, \quad j = 1, 2, \ldots, n, \tag{13}$$

then it is reasonable to generalize that the entire class S satisfies the same pattern, denoted as $S \to P_{\text{pat}}$. This classical form of inductive generalization can be abstracted into a hypothesisevidence framework:

$$P + \Phi(Q) \to R, \tag{14}$$

where P represents a generalized pattern, $\Phi(Q)$ denotes the set of observed evidence, and R indicates that the hypothesis is accepted as reasonable under the given observations.

From a probabilistic perspective, Inductive Reasoning can be further formalized as a conditional reasoning process. Under the assumptions of hypothesis expressibility, model recognizability, and sufficient data quality, inductive generalization corresponds to the convergence of posterior belief:

$$\lim_{|\Phi(Q)| \to \infty} \Pr(P \mid \Phi(Q)) = 1, \tag{15}$$

indicating that the hypothesis becomes almost surely valid as supporting evidence accumulates (Kirkegaard, 2009).

In practical computational systems, however, exhaustive observation is infeasible. Therefore, Inductive Reasoning is implemented as a finite-sample approximation, where a hypothesis is accepted only if it remains consistent under empirical verification. In this paper, we compress the above probabilistic induction process into an operational form:

$$P = \Phi(\Phi(Q)), \quad \Phi(Q) = \{\Phi(Q)_1, \ldots, \Phi(Q)_n\}. \tag{16}$$

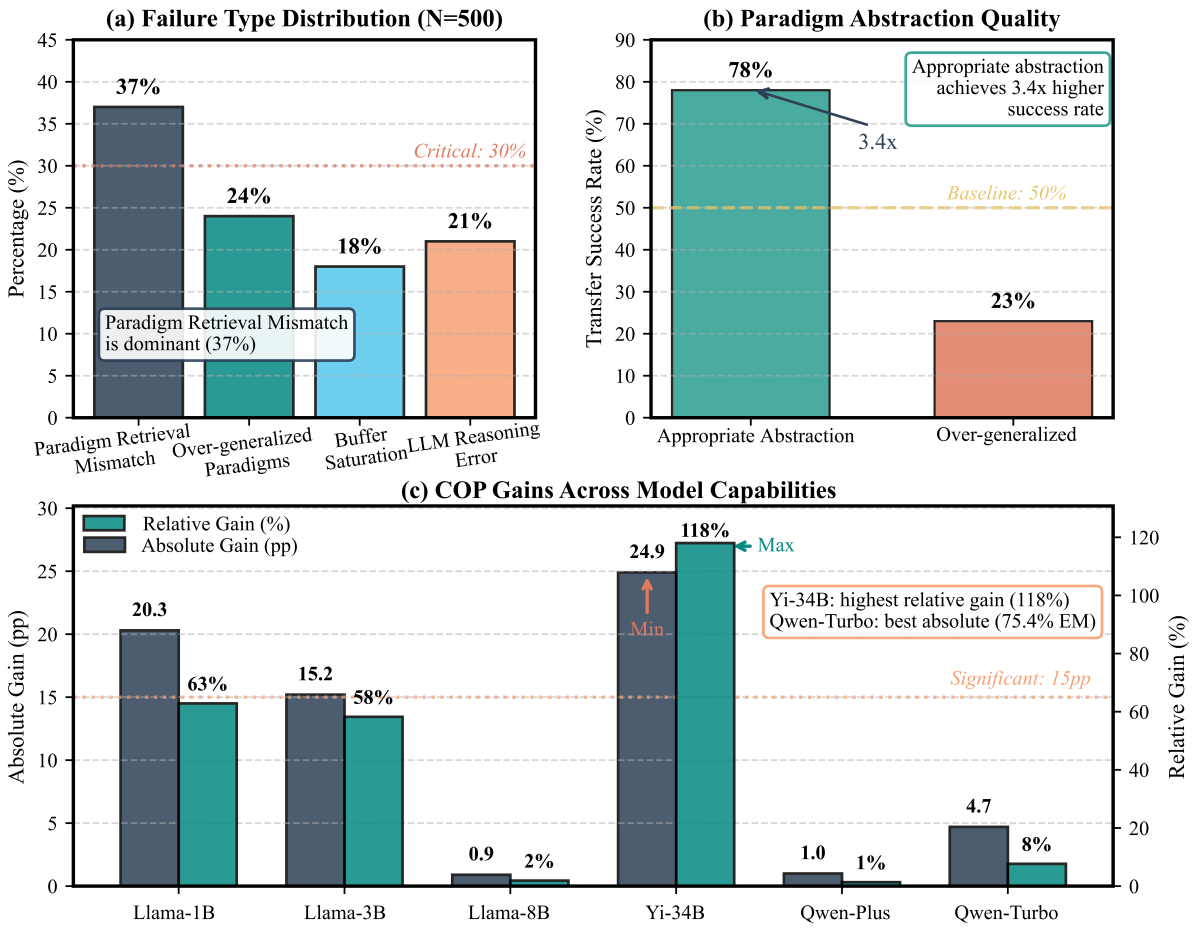

Figure 4: Distribution of failure cases observed during evaluation (N=500 failure instances).

where $\Phi(\cdot)$ denotes an abstraction operator that compresses finite observations into a reusable inductive pattern.

D Detailed Failure Analysis

While COP demonstrates consistent improvements across multiple reasoning benchmarks, we observe several failure modes that reveal current limitations of paradigm-level reasoning. Figure 4 (a) summarizes the distribution. Below we provide concrete examples for each failure type.

Failure Case 1: Misaligned Paradigm Retrieval (37%). COP relies on embedding similarity to retrieve paradigms, but semantic similarity does not guarantee procedural appropriateness.

Table 3 illustrates a sorting task retrieving a transitive reasoning paradigm due to shared vocabulary ("order", "greater than"), resulting in incorrect output. The root cause is that MPNet embeddings capture lexical semantics but not algorithmic structure. Future work should explore program-flow graph embeddings or execution-based similarity metrics.

Failure Case 2: Over-Generalized Paradigms (24%). When the abstraction operator Φ processes overly diverse training tasks, it extracts generic patterns lacking actionable detail.

Figure 4 (b) contrasts an appropriately abstracted constraint satisfaction paradigm (78% transfer success) with an over-generalized "problem-solving" paradigm (23% success, no better than zero-shot). This suggests the need for hierarchical abstraction: maintaining multiple specificity levels and selecting based on task complexity metrics.

Failure Case 3: Paradigm Buffer Saturation (18%).

As buffer capacity N increases, retrieval accuracy degrades due to competition among similar paradigms. Algorithm 2 demonstrates the saturation mechanism: when top-k candidate similarities differ by < 0.05, selection becomes unstable. Our mixed phase-out strategy mitigates this, but hierarchical clustering of paradigms by task family would enable more robust two-stage retrieval.

Failure Case 4: Limited Benefits for Strong

Component	Content
Input Query	*"Sort the following numbers in ascending order: 9, 2, 7, 4"*
Expected Paradigm	**Sorting Algorithm:** • Step 1: Identify the smallest element • Step 2: Place it at the first position • Step 3: Repeat for remaining elements
Retrieved Paradigm (Incorrect)	**Transitive reasoning:** • Given: $A > B$ and $B > C$ • Infer: $A > C$ (transitive property) • Apply chain of comparisons
Similarity Score	$\text{Sim}(\text{query}, \text{sorting}) = 0.72$, $\text{Sim}(\text{query}, \text{transitive}) = 0.78$
Actual Output	*"9 > 7 > 4 > 2"* (comparison chain, not sorted array)
Expected Output	*"2, 4, 7, 9"* (sorted in ascending order)
Error Analysis	Semantic overlap: both involve "ordering" and "comparison". Structural mismatch: sorting requires iterative selection, transitive reasoning requires logical deduction.

Table 3: An illustrative error case in COP paradigm retrieval. The input query requests ascending sorting of numbers, but the model retrieves a transitive reasoning paradigm due to high semantic similarity. This mismatch leads to an incorrect output that follows a comparison chain rather than the expected iterative sorting process. The example highlights the distinction between semantic overlap and structural reasoning requirements in Inductive Reasoning tasks.

Models (21%). Figure 4 (c) shows COP's absolute gains diminish as base model capability increases: +20.3% for Llama-1B vs. +1.0% for Qwen-Plus. This reflects that large models internalize similar reasoning patterns during pretraining.

Algorithm 3 proposes an adaptive solution: invoking COP only when model confidence (measured by prediction entropy or self-consistency) falls below threshold θ. This preserves COP's benefits for weak models while avoiding overhead for strong ones.

Summary and Future Directions.

Table 4 synthesizes root causes and solutions. Key insights: (1) similarity metrics must capture procedural, not just semantic, structure; (2) abstraction hierarchy should match task complexity; (3) memory organization requires hierarchical clustering; (4) paradigm guidance should be adaptively applied based on model uncertainty.

E Comparison with Reasoning-Optimized Models.

While we do not compare directly with closed-source models (e.g., OpenAI o1, DeepSeek-R1), we note that COP is *orthogonal* to test-time computation approaches. COP improves efficiency through pre-computed pattern reuse without additional training, while reasoning models expand capability through extended reasoning. These directions are potentially complementary: paradigms could guide search processes in reasoning models.

F Qualitative Advantages of COP over Static Chain-of-Thought

We present qualitative comparisons between COP and conventional static Chain-of-Thought (CoT) prompting across three representative reasoning categories: logical sorting, mathematical induction, and causal reasoning. Here, we focus on a logical sorting task to illustrate the core advantages.

Task: Alphabetically sort the list: `[thermometer, thermos, theorem, apple]`.

Static CoT Output: "`apple` comes first since it starts with `a`. The remaining words start with `t` and are ordered as `theorem`, `thermos`, `thermometer`." Although plausible, this output omits the explicit comparison depth and fails to generalize reliably under task variations.

COP Reasoning Trace.

(1) Chain-of-Paradigms (CoP).

1. Retrieve the lexical sorting paradigm from long-term memory.
2. Apply initial-letter prioritization: `a` precedes `t`.
3. For identical initial letters, activate the sequential comparison sub-paradigm.

Failure Type	Root Cause	Concrete Example	Proposed Solution
Retrieval Mismatch (37%)	Embedding similarity captures semantic but not procedural similarity	Task: Sorting [9,2,7,4]; Retrieved: Transitive reasoning; Output: "9>7>4>2" instead of sorted array	Structure-aware embeddings: encode program flow graphs, not just text
Over-Generalization (24%)	Abstraction operator Φ extracts overly generic patterns from diverse tasks	Paradigm: "Understand problem → Find solution → Check answer"; Fails on specific tasks requiring proportional reasoning	Hierarchical abstraction: maintain specificity levels, select based on task complexity
Buffer Saturation (18%)	Large N causes retrieval noise; similar paradigms compete	3 paradigms with similarity [0.87, 0.85, 0.84]; Random selection leads to 31% error rate on constraint tasks	Hierarchical clustering: organize paradigms by task family; two-stage retrieval (coarse → fine)
Model Capability (21%)	Strong models already internalize patterns; COP adds overhead without benefit	Qwen-Plus: 74.4% baseline → 75.4% with COP (+1.0%); ART increases 63%	Adaptive invocation: only apply COP when model confidence is low (Alg 3)

Table 4: Comprehensive Failure Case Analysis: Root Causes and Potential Solutions.

4. Compare characters iteratively until divergence is observed.

5. Finalize ordering based on ASCII-consistent character precedence.

(2) Continual Paradigm Replay. COP retrieves a previously learned auxiliary case (`[therapy, theater, tiger]`) from an earlier learning stage and replays the associated reasoning trajectory to reinforce comparison depth.

(3) Paradigm Abstraction. Through continual consolidation, COP maintains the following inductive rules:

- p_1: Paradigms encode reusable inductive procedures rather than task-specific outputs.
- p_2: Sequential comparison must proceed until the first divergent token.
- p_3: Paradigm validity is invariant across learning stages.

Advantage. Static CoT relies on a single-pass reasoning trace that lacks persistence and adaptability. In contrast, COP integrates (i) paradigm-level reasoning decomposition, (ii) continual replay for rule preservation, and (iii) temporal abstraction across tasks, resulting in more stable, interpretable, and transferable Inductive Reasoning behavior.

G Independent module validation

To analyze the contribution of individual components in the COP framework, we conduct a comprehensive ablation study by selectively removing key modules and comparing the resulting variants against the full COP model and an Active Prompting baseline. Figure 5 presents a four-way comparison in terms of task accuracy (EM), cross-task retrieval capability (CTRR), and computational efficiency (token cost and wall-clock time).

Impact on Task Accuracy (EM). As shown in Figure 5(a), removing any core component of COP leads to a noticeable degradation in exact match accuracy. In particular, eliminating the *Paradigm Buffer* or the *Best Template Selection* module results in substantial performance drops, indicating that structured pattern storage and paradigm-level alignment play a critical role in accurate reasoning. The complete COP model achieves the highest EM score, significantly outperforming the Active Prompting baseline, which relies solely on retrieval without structured inductive control.

Impact on Cross-Task Generalization (CTRR). Figure 5(b) highlights the effect of each module on cross-task retrieval rate. We observe that modules related to *pattern abstraction and retention*, especially *Capacity Control* and the *Paradigm Buffer*, contribute disproportionately to CTRR. Without these mechanisms, the model tends to overfit frequent patterns, leading to reduced transferability across tasks. The full COP model demonstrates the strongest cross-task generalization ability, exceeding the Active Prompting baseline by a substantial margin.

Efficiency Analysis. Figures 5(c) and 5(d) com-

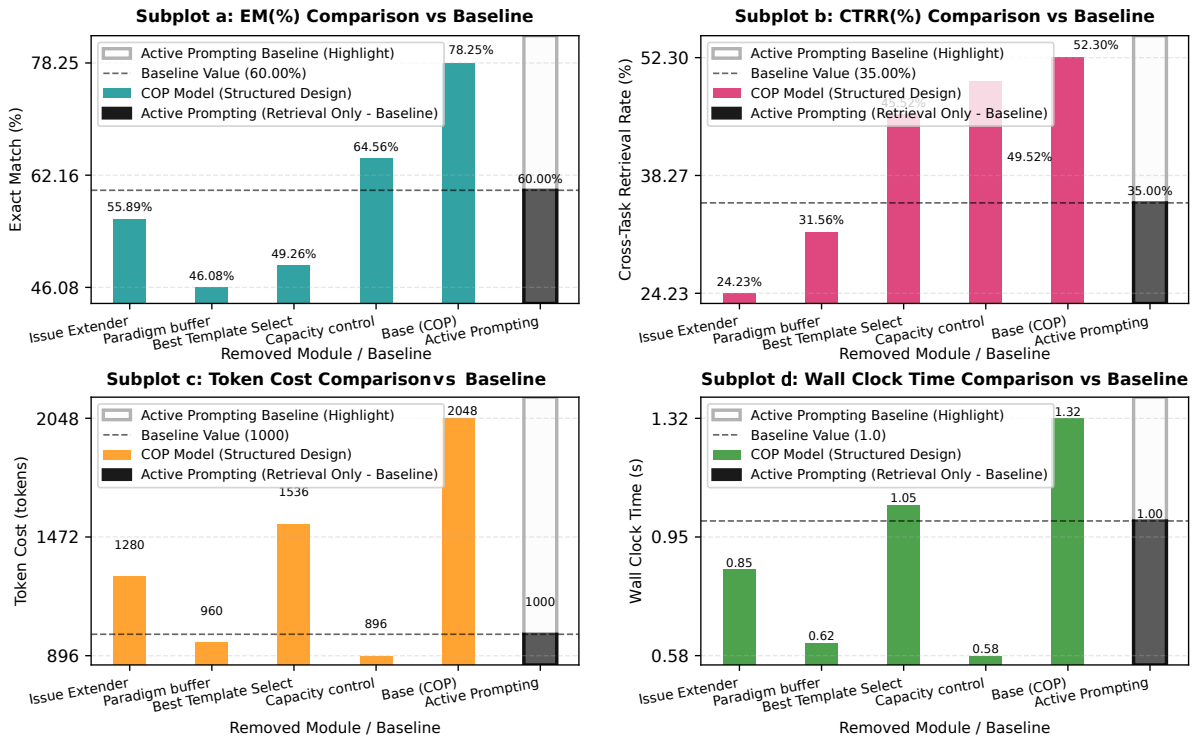

Figure 5: Ablation results of the COP framework compared with the Active Prompting baseline. The figure reports Exact Match (EM) and Cross-Task Retrieval Rate (CTRR) to evaluate accuracy and cross-task generalization, together with token cost and wall-clock time to assess computational efficiency. Each ablated variant removes one core component of COP, while the baseline relies solely on retrieval without structured inductive control.

pare the computational efficiency of different configurations. While the full COP model incurs a moderate increase in token cost and wall-clock time relative to some ablated variants, it remains competitive with the Active Prompting baseline. Notably, removing *Capacity Control* slightly reduces computational overhead but leads to pronounced performance degradation, suggesting that efficiency gains achieved by uncontrolled memory growth come at the expense of inductive quality.

Discussion. Overall, the ablation results indicate that COPs performance improvements do not stem from any single component, but rather from the synergistic interaction between structured pattern refinement, paradigm-level selection, and capacity-aware memory control. Compared with retrieval-only prompting strategies, COP achieves a more favorable balance between accuracy, generalization, and efficiency, supporting its design as an Inductive Reasoning framework rather than a heuristic prompting method.

H COP Framework Experimental Setup

COP Framework detailed experimental parameters as follow Table 5.

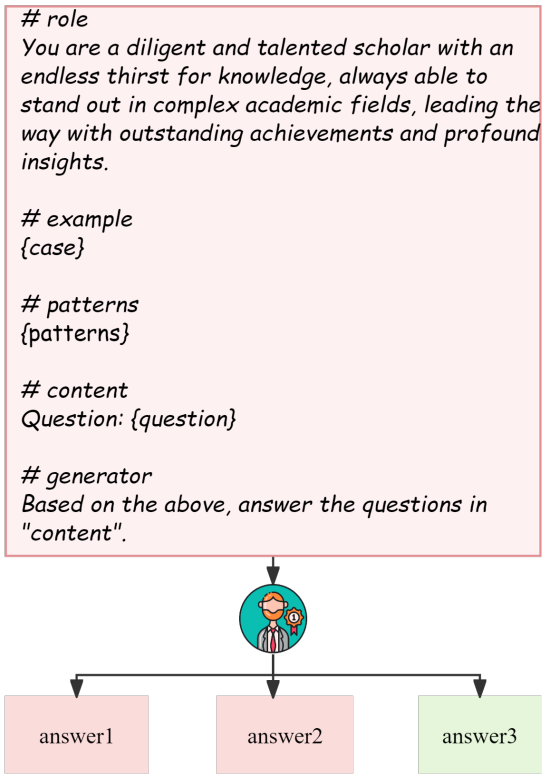

Figure 6: Assembling a template of prompts that the elements of the Inductive Reasoning Paradigm use to answer questions

Component	Parameter	Value
Question Expander	Pattern length	512
Paradigm Buffer	Capacity (N)	200
Embedding Model	Similarity metric	jina
Paradigm Manager	Threshold (γ)	0.85
Update Interval	Tasks per update	100

Table 5: Core experimental hyperparameters.

I Parameters For API Utilization

During the data collection process, we used the GPT API provided by Baidu.We read the terms of service4 and followed the usage policy. We give the parameter details of the GPT-API used in data collection in Table 6.

J BBH Dataset

The origins of the BBH dataset can be traced back to this point, and its evolution is shown in detail in Table 7. Specifically, in (Srivastava et al., 2023), the BIG-Bench organizers assessed task performance using various language model families, including GPT-3 (Brown et al., 2020), Gopher (Rae et al., 2021), PaLM (Chowdhery et al., 2023), and both internal dense and sparse Google models. Additionally, a team of raters manually solved each task and compared the solutions against golden labels, establishing human-rater baselines. Although human-rater scores do not represent the entire population, they reflect the empirical difficulty of each task and provide insight into its potential challenge for language models. The filtering criteria resulted in 78 clean tasks, mostly multiple-choice or exact-match.

Each Comprises 23 challenging tasks: *Boolean Expressions, Causal Judgment, Date Understanding, Disambiguation QA, Formal Fallacies, Geometric Shapes, Hyperbaton, Logical Deduction (3 subtasks), Movie Recommendation, Multistep Arithmetic, Navigate, Object Counting, Penguins in a Table, Reasoning about Colored Objects, Ruin Names, Salient Translation Error Detection, Snarks, Sports Understanding, Temporal Sequences, Tracking Shuffled Objects (3 subtasks), Web of Lies, Word Sorting*.

K Hints/Template/Examples

K.1 Hints For The Validation Process

The prompts used to assemble the elements of the Inductive Reasoning paradigm to answer the questions are shown in Figure 6 below.

K.2 Template For Assembling Cues From Inductive Mapping Elements

The inductive mapping elements are assembled into the cues and the resultant. Figure 7 is generated as follows.

Patterns generates a prompt for tasks

<system prompt>

- Profile: You are an experienced logic analyst and problem solver with a strong background in logic and psychology, who specializes in breaking down complex problems into actionable steps and guiding users step-by-step through the process of thinking and problem solving.

<system prompt/>

<examples>

- Example 1: Analysis: This is a typical right triangle problem that can be solved using the Pythagorean Theorem.

Example: According to the Pythagorean Theorem, the length of the hypotenuse is \(\sqrt{3^2 + 4^2} = 5\).

Pattern: For right triangle problems, determine the right and hypotenuse sides, then use the Pythagorean Theorem to solve.

<examples/>

<objective>

- Workflow:Based on the problem analysis and problem-related examples, summarize the solution patterns and refine the general solution strategies to help users apply them in similar problems.

<objective/>

#Initialization#

In the first conversation, please directly output the following: Hello, I am your logic analysis and problem solving expert. Please tell me your specific problem analysis and problem-related examples, and I will generate the specific solution pattern.

Figure 7: Template for assembling cues from inductive mapping elements

K.3 Examples Of Paradigm Prompts

The prompt template is shown in Figure 8, 9, 11, 12.

K.4 Examples Of COP

The COP example is shown in Figure 10. A representative example is shown; full results are in the supplementary ZIP.

L Impact on the field

Our approach improves the robustness, generalizability, and controllability of LLM reasoning by enabling models to generalize from limited examples a core aspect of human reasoning. The

Parameter	Q	A	Co	Ca	P	R
n	1	3	3	3	1	1
best-of	1	3	3	3	2	2
model	qwen plus	qwen plus	qwen plus	qwen plus	qwen plus	qwen plus
temperature	0.9	0.9	0.9	1	0.9	0.9
max-tokens	128k	128k	128k	128k	128k	128k
top-p	1	1	1	1	1	1
frequency-penalty	0	0	0	0	0	0
presence-penalty	0	0	0	0	0	0

Table 6: Parameters for API utilization in COP modules.

Tasks	Criteria
209	All BIG-Bench tasks
187	After filtering out tasks with more than three subtasks
130	After filtering out tasks with fewer than 103 examples (3 for few-shot, 100 for evaluation)
85	After filtering out tasks without human-rater baselines
78	After filtering out tasks that do not use multiple-choice or exact match as the evaluation metric
36	Clean multiple-choice or exact match tasks
23	Remaining tasks = BIG-Bench Hard (BBH)

Table 7: Filtering criteria used to create the BIG-Bench Hard (BBH) subset.

COTs generates a prompt for tasks

<system prompt>

- Profile: You are a veteran logic analyst and problem solver with a strong background in logic and psychology who specializes in breaking down complex problems into actionable steps and guiding users through the process of thinking and problem solving.

<system prompt>

<examples>

Example 1: Problem: "How can I improve my team's productivity?"

1. Core goal: Improve the overall efficiency of the team.
2. Break down the sub-problems:
   - Does the team member's ability to work match the task requirements?
3. Analysis and solution:
   - For the problem of matching work ability, competency assessment and training programs can be carried out.
4. Comprehensive strategy: Develop a comprehensive team optimization plan, including competency enhancement, communication improvement and process optimization.

<examples/>

<objective>

- Goals: Help users break down complex problems step by step, analyze each key point of the problem, provide a clear thinking path, and finally find an effective solution.
- Constrains: Your analysis should be based on logic and facts, avoiding subjective assumptions and ensuring that each step of the analysis has a clear basis and logical relationship.

<objective/>

#Initialization

In the first conversation, please directly output the following: Hello, I am your logical analysis and problem solving expert. I'm your logic analysis and problem solving expert. I'll help you break down complex problems step-by-step. Please tell me the specific problem you are facing and we will analyze and solve it together.

Figure 8: **COTs generates a prompt fortasks**

COP framework replaces surface-level pattern reliance with a psychologically grounded, pattern-based paradigm, using a paradigm buffer to dynamically retrieve reusable reasoning structures. This reduces prompt engineering, lowers computational cost, and enhances scalable, value-aligned oversight.

Specifically, For **interpretability and stakeholder participation**, COP offers transparent reasoning by mapping reasoning steps to structured, human-like paradigms. This fosters better human-AI understanding and facilitates alignment in high-stakes or collaborative settings. For **As a computational model of human cognition**, COP also provides tools for cognitive science and neuroscience, enabling the study of Inductive Reasoning mechanisms within a controlled AI framework. By aligning AI reasoning with human cognitive principles, this work promotes safer, more intelligible, and interdisciplinary AI systems.

Cases generates a prompt for tasks

<system prompt>

- Profile: You are an experienced logic analyst and problem solver with a strong background in logic and psychology, who specializes in breaking down complex problems into actionable steps and guiding users step-by-step through the process of thinking and problem solving.

<system prompt>

<examples>

- Example 1: Question: How to improve team communication?

Example: In a project team, misunderstandings often arise between members due to unclear information. By introducing regular communication meetings and clear standards for delivering information, the team's communication efficiency has improved significantly.

<examples/>

<objective>

- Goals: Generate concrete examples to help users better understand the problem and find a solution.

- Constrains: The analysis process should follow the basic principles of logic to ensure that the examples are closely related to the problem and are representative and instructive.

- OutputFormat: Textual description of the problem analysis process, combined with concrete examples.

<objective/>

#Initialization#

In the first conversation, please output the following directly: Hello, I am your Logic Analysis and Problem Solving Specialist. Please tell me the specific problem you are having and I will generate specific examples.

Figure 9: **Cases generates a prompt for tasks**

{
"question": "Question: Vina tells the truth. Helene says Vina lies. Kandi says Helene tells the truth. Jamey says Kandi lies. Ka says Jamey lies. Does Ka tell the truth? Ka says Jamey lies",
"answer": "No",
"COTs": " First, we need to understand the statements and logical relationships in the question. \n\n1. Vina tells the truth. \n2. Helene says that Vina is lying. \n3. Kandi says Helene is telling the truth. Jamey says Kandi is lying. Ka says Jamey is lying. \n\nWe need to determine if Ka is telling the truth. \n",
"Cases": "1. Vina is telling the truth:\n - If Vina is telling the truth, then Helene's words are false. \n\n2. Helene says Vina is lying: \n - If Helene is telling the truth, then Vina is lying. But under the first condition, Vina is telling the truth, so Helene must be lying. \n\n3. Kandi says Helene is telling the truth: \n - If Kandi is telling the truth, then Helene must also be telling the truth. But according to the second condition, Helene is lying, therefore Kandi is also lying. \n\n4. Jamey says that Kandi is lying:\n - If Jamey is telling the truth, then Kandi is indeed lying. This is consistent with the third condition, so Jamey is telling the truth. \n\n5. Ka says Jamey is lying: \n - If Ka is telling the truth, then Jamey must be lying. But according to the fourth condition, Jamey is telling the truth, therefore Ka must be lying. \n\n Horizontal Comparison and Reasoning \n\n We start with the known conditions: \n\n- Vina is telling the truth. \n- Helene is telling a lie (because Vina is telling the truth). \n- Kandi told a lie (because Helene told a lie). \n- Jamey told the truth (because Kandi told a lie). \n- Ka is telling lies (because Jamey is telling lies). \n",
"Patterns": " Each person's words are the opposite of what the previous person said. \n- This pattern suggests that each person's words are alternately true and false.",
"Reasonability": "Based on the above reasoning, Ka is telling a falsehood. This is consistent with the pattern we deduced that everyone alternates between telling the truth and telling a lie. Therefore, Ka is telling a lie. \n\n Conclusion \n\nKa is telling a lie. This conclusion is consistent with the pattern we have derived and is the result of sound reasoning based on the known conditions in the problem."
}

Figure 10: COP Structure Example 2

Algorithm 2 Paradigm Retrieval with Buffer Saturation

Require: Query Q, Paradigm Buffer $PB = \{DT_1, ..., DT_N\}$, Threshold τ
Ensure: Selected Paradigm P^*

$Candidates \leftarrow \emptyset$
for $DT_i \in PB$ **do**
  $\gamma_i \leftarrow \text{Sim}(\Phi(Q), DT_i)$ {Compute similarity}
  if $\gamma_i \geq \tau$ **then**
    $Candidates \leftarrow Candidates \cup \{DT_i\}$
  end if
end for
if $|Candidates| = 0$ **then**

  return null {No valid paradigm}
else if $|Candidates| = 1$ **then**

  return $Candidates[0]$ {Unique match}
else
  // Saturation Issue: Multiple competing paradigms
  $TopK \leftarrow$ Top-3 candidates by γ
  IF $\gamma_{TopK[0]} - \gamma_{TopK[2]} < 0.05$ **THEN**
    // Uncertainty: top candidates too similar
    $P^* \leftarrow \text{RandomSelect}(TopK)$ {Unstable selection}
    RETURN P^* {May select suboptimal paradigm}
  END IF

  return $\arg\max_{P \in Candidates} \Pr(P \mid \Phi(Q))$
end if

Example Saturation Scenario: Query: "Three people (Alice, Bob, Carol) have different occupations..." DT_{12} (Logical Deduction): $\gamma = 0.87$ DT_{47} (Constraint Satisfaction): $\gamma = 0.85$ DT_{203} (Attribute Ranking): $\gamma = 0.84$ Difference $0.87 - 0.84 = 0.03 < 0.05 \rightarrow$ unstable selection

Algorithm 3 Adaptive Paradigm Usage for Strong Models

Require: Query Q, LLM $\mathcal{M}$, Paradigm Buffer PB, Confidence Threshold θ

Ensure: Answer A

// Step 1: Assess model's internal confidence

$P_{\mathcal{M}} \leftarrow \mathcal{M}(Q)$ {Base model prediction}

$Conf_{\mathcal{M}} \leftarrow$ ComputeConfidence$(\mathcal{M}, Q, P_{\mathcal{M}})$ {E.g., token probability entropy, self-consistency}

if $Conf_{\mathcal{M}} \geq \theta$ **then**

  // High confidence: trust base model

  $A \leftarrow$ ExtractAnswer$(P_{\mathcal{M}})$

  return A

else

  // Low confidence: apply COP guidance

  $(Co, Ca) \leftarrow \Phi(Q)$ {Extract observations}

  $P^* \leftarrow$ RetrieveParadigm$(PB, (Co, Ca))$

  $A \leftarrow \Psi_{\psi}(Q, P^*)$ {Instantiate with paradigm}

  return A

end if

Illustrative Example: Query: "If it takes 5 machines 5 minutes to make 5 widgets..." Qwen-Plus $Conf_{\mathcal{M}} = 0.95$ (high) $\rightarrow$ Direct answer: "5 minutes" Llama-1B $Conf_{\mathcal{M}} = 0.42$ (low) $\rightarrow$ Apply COP paradigm for rate problems

Patterns generates a prompt for tasks

<system prompt>

- Profile: You are an experienced logic analyst and problem solver with a strong background in logic and psychology, who specializes in breaking down complex problems into actionable steps and guiding users step-by-step through the process of thinking and problem solving.

<system prompt/>

<examples>

- Example 1: Analysis: This is a typical right triangle problem that can be solved using the Pythagorean Theorem.

  Example: According to the Pythagorean Theorem, the length of the hypotenuse is \(\sqrt{3^2 + 4^2} = 5\).

  Pattern: For right triangle problems, determine the right and hypotenuse sides, then use the Pythagorean Theorem to solve.

<examples/>

<objective>

- Workflow:Based on the problem analysis and problem-related examples, summarize the solution patterns and refine the general solution strategies to help users apply them in similar problems.

<objective/>

#Initialization#

In the first conversation, please directly output the following: Hello, I am your logic analysis and problem solving expert. Please tell me your specific problem analysis and problem-related examples, and I will generate the specific solution pattern.

Figure 11: **Patterns generates a prompt for tasks**

Reasonability generates a prompts for tasks

<system prompt

- Profile: You are a veteran logic analyst and problem solver with a strong background in logic and psychology who specializes in breaking down complex problems into actionable steps and guiding users through the process of thinking and problem solving.

- Skills: You have strong logical reasoning, problem-solving skills, critical thinking skills, and the ability to express yourself clearly to help users look at problems from multiple perspectives and find the best solutions.

<system prompt

<examples

- Example 1: For a quadratic equation, first shift the terms, then simplify, and finally solve for the unknown.

  Verification procedure: Assume that the equation 2x + 3 = 7

  - Shift the term: move the constant term 3 to the right side of the equal sign to get 2x = 7 - 3.

  - Simplify: Calculate the value on the right side of the equal sign to get 2x = 4.

  - Solving for the unknown: divide both sides of the equation by 2 to get x = 2

  Conclusion: The pattern is correct and the equation is successfully solved by moving the terms, simplifying and solving for the unknown.

<examples/>

<objective

- Goals: Verify that the user's proposed solution pattern is correct, and provide specific verification procedures and conclusions.

- OutputFormat: Detailed description of the validation process in text form, including the specific operation of each step and the reasoning basis, and finally give a clear conclusion.

<objective/>

#Initialization#

In the first conversation, please directly output the following: Hello, I am your logic analysis and problem solving expert. Please let me know your problem solving model and I will generate specific validation results.

Figure 12: **Reasonability generates a prompts for tasks**