MAPPING BEFORE GENERATING: IMAP FOR INDUCTIVE REASONING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models excel at deductive reasoning and context learning, but their inductive reasoning capabilities remain limited by the fact that the models tend to fit instance-level answers rather than abstract reusable general patterns from a finite set of examples. This shortcoming severely hampers the generation of scientific hypotheses, cross-domain generalization, and efficient learning from samples. We propose IMAP, a reinforcement learning-based intelligent mapping framework that integrates the inductive reasoning paradigm into model inference. We designed a paradigm mapping structure comprising four core elements: chains of thought (CoT), cases, patterns, and rationality. We also propose the RL Paradigm Model (RLP) for paradigm generation. By using inductive reasoning as an input cue, we guided multiple large models to generate an average of 270 results. Comparative experiments demonstrate that combining input cues with inductive reasoning performs well across most models and enhances the quality of generated results. By framing inductive reasoning as a learnable paradigm-mapping process, our work unlocks its generative potential and offers a new perspective for understanding and deconstructing the reasoning mechanisms of large language models.[1]

1 INTRODUCTION

A core limitation of current large language models (LLMs) is that they tend to be suited for instance-level answers rather than autonomously inducting abstract, reusable, and general patterns from a finite set of examples. This limitation severely hinders the generation of scientific hypotheses, cross-domain generalization, and efficient learning from samples. Currently, mainstream techniques such as Chain-of-Thought (CoT) prompts and Reinforcement Learning with Human Feedback (RLHF) typically treat CoT as the overall trajectory of policy optimization, without explicit knowledge abstraction. Therefore, we believe that inductive reasoning should be constructed as a learnable mapping process from examples to reusable patterns, rather than a byproduct of answer fitting.

To bridge this gap, we propose **I**nductive **M**apping b**a**sed on **P**aradigm (IMAP), a framework that embeds a structured, self-evolving inductive reasoning process within LLM inference. Specifically, the process of self evolution consists of a dynamic multi-level knowledge architecture that includes a chain of thought (CoT), cases, patterns, and reasonability, as well as a closed-loop algorithm that connects them to each other. This data structure is dynamically optimized through a composite reward reinforcement learning (RL) mechanism with adaptive constraints, enabling it to effectively promote alignment between the internal representations of the model and human style inductive logic.

Based on this framework, our contributions are as follows. First, the **inductive mapping** mechanism extracts reusable *Cases* and general *patterns* from the initial CoT, creating a dynamic, hierarchical reasoning graph. Second, an integrated **self-verification** module (*plausibility*) provides soft consistency feedback for the generated knowledge structure, reducing reliance on opaque external reward models. Finally, the model structure is continuously improved through reinforcement learning guided by an adaptive KL controller.

[1] https://anonymous.4open.science/r/IMAP-inductive-LLM-FB5B

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

As large language models (LLMs) become essential for applications like conversational agents, content generation, and problem-solving systems, their ability to perform inductive reasoning is under greater scrutiny Latona et al. (2024); Heiding et al. (2023); Dam et al. (2024). However, the core issue is not merely whether LLMs can provide correct answers, but rather whether they can autonomously learn to extract abstract and reusable general patterns from a limited number of examples. In particular, this issue is directly related to the generation of scientific hypotheses, cross-domain generalization, and efficient learning from samples.

Numerous studies have explored the inductive capabilities of large language models. Examples include step-by-step reasoning methods such as Chain of Thought (CoT) Zhang et al. (2024b), tree-structured CoT Yao et al. (2023), reinforcement learning with human feedback (RLHF), and reasoning-oriented derivative methods (e.g., Long et al. (2024); Zhang et al. (2024a)). However, these classical approaches have several issues:

Sparse and Static Reasoning Annotations: Standard CoT-based methods typically use a single annotated reasoning path per question Zhang et al. (2024b). This static approach fails to capture the space of multiple valid reasoning strategies, constraining the model's ability to adapt its reasoning flexibly.

Passive Knowledge Structuring: Although a structured thinking chain can organize the steps of reasoning, the reasoning process itself remains merely a trajectory and cannot evolve into a self-improving knowledge base.

Scalability and generalization in RL-based methods. RL frameworks such as RAFT Zhang et al. (2024a) show promise but scale poorly: complex reasoning yields only sparse rewards, and effective training often relies on domain-specific datasets for reward shaping. Recent abstraction-based RL methods, including AbstRaL Gao et al. (2025), Abstraction-of-Thought Hong et al. (2024), and Logic-RL Xie et al. (2025), alleviate reward sparsity through abstraction, but they (i) represent abstractions as discrete abstraction, (ii) depend on hard external verifiers to shape rewards, and (iii) assume stationary reward signals, an assumption that breaks once the underlying knowledge base evolves.

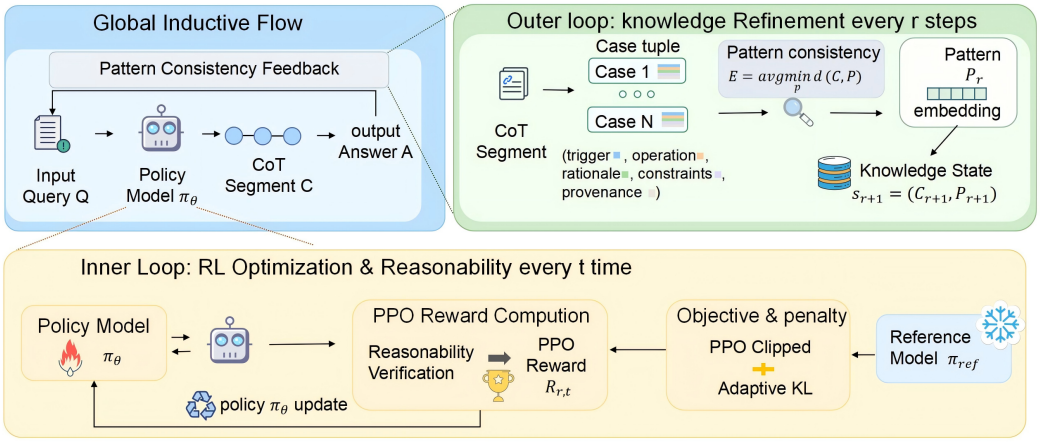


Figure 1: Overview of the IMAP inductive framework pipeline. **Inner loop (every t steps):** The policy model π_θ generates a chain-of-thought (CoT) segment C for an input query Q, which is verified for reasonability and scored by a PPO reward $R_{r,t}$; the clipped PPO objective with adaptive KL divergence against the frozen reference model π_{ref} then updates π_θ. **Outer loop (every r steps):** The accumulated CoT segments are organized into case tuples and distilled into a pattern set $\mathcal{P}_r$; pattern consistency is measured as $E = \text{avgmin}\, d(C, \mathcal{P})$, yielding an updated knowledge state $s_{t+1} = (C_{t+1}, \mathcal{P}_{r+1})$ that feeds back as global inductive bias to the policy. At test time, π_θ and the pattern state s_r are frozen to produce the final answer A.

Inspired by the issues mentioned above, this paper proposes IMAP. Its core design features include. First, knowledge Representation storing cases and patterns as complete text-based knowledge units, making the extracted knowledge viewable, reusable, and combinable. Second, Reward Construction using soft rationality feedback to construct rewards, thereby eliminating the reliance on rigid external validators or domain-specific datasets. Third, Training Stability adaptively adjusting the KL penalty term in response to non-stationary reward signals caused by an evolving knowledge base, thereby ensuring stable training in reward drift scenarios.

3 IMAP FRAMEWORK

3.1 IMAP FRAMEWORK OVERVIEW

The IMAP framework presents reasoning as an iterative, structured inductive process that transforms an input query Q into an answer A through a hierarchy of structured, inspectable abstractions, namely Cases and Patterns. The abstract flow is as follows:

$$Q \to \mathrm{CoT} \to \mathrm{Cases} \to \mathrm{Patterns} \to A, \qquad \mathrm{Patterns} \to \mathrm{CoT\ (feedback)}, \tag{1}$$

where the arrows denote abstraction levels and information flow rather than a strictly one-pass sequence. The feedback arrow indicates that evolving Cases and Patterns influence subsequent CoT generation through retrieval and reward computation through the Reasonability module (Sec. 3.3).

As illustrated in Figure 1, this process is instantiated as a cyclical structure that includes CoT segment generation, case induction, pattern abstraction, and rationality verification. Unlike standard Chain of Thought (CoT) Hsieh et al. (2023) combined with reinforcement learning from human feedback (RLHF) Ouyang et al. (2022), IMAP treats reasoning trajectories not merely as supervisory signals, but as structured inputs for iterative abstraction and rule extraction. The strategy for generating reasoning trajectories is optimized by RL with a composite reward and an adaptive KL controller Schulman et al. (2017b) (Eq. equation 11), while Cases and Patterns are periodically updated separately from the policy parameters by a set of LLM-based induction operators.

The complete procedures are formalized in App. F.4 and App. F.7. We unify the indices as follows: r indexes knowledge-update rounds, k indexes PPO update steps, and t indexes reasoning steps. The unified notation is summarized in Table 6; detailed tuple definitions and consistency conventions are deferred to Appendix F.3.

3.2 STRUCTURED KNOWLEDGE STATE AND NOTATION

IMAP maintains an explicit, evolving knowledge state $s_r = (\mathcal{C}_r, \mathcal{P}_r)$, where $\mathcal{C}_r$ and $\mathcal{P}_r$ denote the structured case set and the textual Pattern set at knowledge-update round r, respectively. We initialize $\mathcal{C}_0 = \emptyset$ and $\mathcal{P}_0 = \emptyset$. A set of LLM-based induction operators, used separately from the policy parameters, periodically updates this state (Section 3.3). These operations include, among others, the case induction operator and the pattern abstraction operator.

The structural design for cases C and paradigms P draws on the structured case representation Bergmann et al. (2005) used in case-based reasoning, the trigger condition and constraint design Aamodt & Plaza (1994) used in pattern driven reasoning, and the embedded rule-based retrieval requirements in the LLM era. as follows:

Cases are stored as structured tuples

$$c = (\text{trigger}, \text{operation}, \text{rationale}, \text{constraints}, \text{provenance}),$$

where provenance records the source query and the CoT segment from which the case is induced. Patterns are stored as textual rules

$$p = (\text{trigger condition}, \text{reasoning schema}, \text{applicability constraints}, \text{support count}, \text{embedding}),$$

where support count is updated from the number of assigned cases, and the embedding is computed by the fixed sentence encoder.

3.3 CORE MODULES AND INDUCTIVE MAPPING

Given Question Q and s_r, the policy π_θ generates reasoning segments. At each round, the policy conditions on Q and the retrieved $top-k$ Patterns from $\mathcal{P}_r$ to generate CoT_r, where $top-k=5$ by default; case induction operator decomposes this trace into semantic segments and extracts structured cases; pattern abstraction operator optimizes soft case-to-Pattern assignments and then generates textual rules; reasonability verification computes an energy-based consistency score LeCun et al. (2006). The complete procedure is formalized in Algorithm 2, while policy optimization is formalized separately in Algorithm 3. We describe the core modules below.

Case induction operator. Given a sampled reasoning trace CoT_r, IMAP decomposes it into semantic fragments $\{z_i\}$, and after ambiguity determination, g_{case} converts each fragment into a structured case. Each case follows the tuple format $c = (\text{trigger}, \text{operation}, \text{rationale}, \text{constraints}, \text{provenance})$, where provenance records the source query and the CoT segment from which the case is induced. The specific criteria for determining semantic ambiguity can be found in Appendix F.1.

Then, if no ambiguity occurs, case induction is deterministic, and we use g_{case} to map each segment into a structured tuple. Specifically, g_{case} operates as a two-stage deterministic parser: first, a rule-based splitter identifies candidate spans using discourse markers and clause boundaries; second, an LLM-based semantic parser at temperature 0 extracts the five tuple fields by filling a fixed prompt template, so that the same segment always yields the same tuple.

However, if ambiguity occurs, we instantiate the stochastic kernel $p_{\phi_{\text{case}}}(\mathcal{C}_{r+1} \mid \mathrm{CoT}_r, \mathcal{C}_r)$, and we either marginalize over valid segmentations. The logical update is

$$\mathcal{C}_{r+1} = \mathcal{C}_r \cup \{g_{\text{case}}(z_i) \mid z_i \in \mathrm{Segments}(\mathrm{CoT}_r)\}, \tag{2}$$

$\mathrm{Segments}(\cdot)$ is a rule-based splitter followed by an LLM-based semantic parser, and each field in the tuple is stored as a short natural-language span.

The induced cases are accumulated into $\mathcal{C}_{r+1} = \mathcal{C}_r \cup \{g_{\text{case}}(z_i)\}$, which provides structured evidence for subsequent Pattern abstraction.

Pattern abstraction operator. Given the accumulated cases $\mathcal{C}_{r+1}$, IMAP first estimates case embeddings and optimizes the auxiliary assignment parameters ψ by maximizing a regularized soft-assignment likelihood over existing Patterns. The pattern likelihood is

$$P_\psi(p \mid c) = \frac{\exp(\mathrm{sim}(e_c, f_\psi(p))/T_{\text{sim}})}{\sum_{p' \in \mathcal{P}_r} \exp(\mathrm{sim}(e_c, f_\psi(p'))/T_{\text{sim}})}, \quad \mathcal{P}_r \neq \emptyset, \tag{3}$$

where T_{sim} is the soft-assignment temperature, $\mathrm{sim}(a,b) = \cos(\mathrm{Emb}(a), \mathrm{Emb}(b))$ denotes cosine similarity between embeddings, $e_c = \mathrm{Emb}(c)$ is a fixed case embedding computed by a sentence encoder (all-MiniLM-L6-v2), and $f_\psi(p)$ is initialized from the same fixed sentence encoder applied to the textual pattern p, and ψ only adjusts the embedding through a low-rank linear projection. Here, $f_\psi(p')$ denotes the same pattern embedding function $f_\psi(\cdot)$ applied to another candidate pattern $p' \in \mathcal{P}_r$.

If $\mathcal{P}_r = \emptyset$, we skip the soft-assignment likelihood in Eq. equation 3 and instead bootstrap an initial pattern set $\mathcal{P}_{r+1}$ directly from $\mathcal{C}_{r+1}$ using the LLM-based pattern-abstraction operator. However, semantic similarity does not guarantee logical coverage, and frequent patterns may dominate low-frequency but critical constraints. Therefore, we discuss failure cases in Section 4.5.

When $\mathcal{P}_r \neq \emptyset$, to extract reusable schemas, we optimize the auxiliary pattern-assignment parameters ψ by a regularized maximum-likelihood objective. Let $q_r(p \mid c)$ be the soft assignment from the E-step, e.g.,

$$q_r(p \mid c) = P_{\psi_r}(p \mid c).$$

Then, the M-step updates ψ by the following regularized soft-assignment objective:

$$\psi_{r+1} = \arg\max_{\psi} \mathbb{E}_{c \sim \mathcal{C}_{r+1}} \mathbb{E}_{p \sim q_r(\cdot \mid c)} [\log P_\psi(p \mid c)] - \lambda\, \Omega(\psi), \tag{4}$$

where $\lambda_{\text{reg}} > 0$ is the regularization coefficient, $P_\psi(p \mid c)$ is defined in Eq. equation 3 and measures how well pattern p explains an observed case under the soft-assignment embedding $f_\psi(p)$, and $\Omega(\psi)$

penalizes overly complex assignments or representations. Specifically, we set $\Omega(\psi) = \|\psi\|_F^2 + \alpha \cdot |\{p : \sum_c q_r(p \mid c) > 0\}|$, where $\alpha > 0$ balances the two terms; the first term controls embedding norm and the second term discourages activating too many patterns. During the M-step, the pattern set $\mathcal{P}_r$ in Eq. equation 3 is held fixed; therefore, only ψ is optimized. After the LLM operator produces $\mathcal{P}_{r+1}$, the likelihood is re-instantiated with $\mathcal{P}_{r+1}$.

Reasonability verification. The Reasonability module measures how well a reasoning trace aligns with the current Pattern set. It computes an energy

$$E(\mathrm{CoT}_{r,\le t}, \mathcal{P}_r) = \frac{\sum_{z_i \in \mathrm{Segments}(\mathrm{CoT}_{r,\le t})} \min_{p \in \mathcal{P}_r} d\big(g_{\mathrm{case}}(z_i), p\big)}{\max(1, |\mathrm{Segments}(\mathrm{CoT}_{r,\le t})|)}. \tag{5}$$

Let $u = g_{\mathrm{case}}(z_i)$, where $d(u, p)$ combines node similarity, edge similarity, order violation, and constraint matching. Specifically, The calculation process of the discrepancy $d(u, p)$ is detailed in Appendix F.2.

For $\mathcal{P}_r \neq \emptyset$, the resulting consistency score is $C_{r,t} = \exp(-E)$, and the verification flag is $\mathrm{flag}_{r,t} = \mathbb{1}(C_{r,t} \ge \tau)$, with threshold $\tau = 0.75$. If $\mathcal{P}_r = \emptyset$, we set $C_{r,t} = 0$ and $\mathrm{flag}_{r,t} = 1$ as neutral values to disable the consistency reward and the penalty. This signal is incorporated into the reward in Section 3.4. The penalty term in Eq. equation 6 is active only when $\mathcal{P}_r \neq \emptyset$.

3.4 RL Optimization with Inductive Paradigm Rewards

The inconsistency signal from the Reasonability module (Sec. 3.3) is incorporated into the reward, and the policy is updated by PPO Schulman et al. (2017b) under structurally informed rewards. At each reasoning step t in round r, the reward is

$$R_{r,t} = w_{\mathrm{acc}} R_{r,t}^{\mathrm{acc}} + w_{\mathrm{struct}} R_{r,t}^{\mathrm{struct}} + w_{\mathrm{consist}} R_{r,t}^{\mathrm{consist}} - w_{\mathrm{pen}} \mathbb{1}(\mathrm{flag}_{r,t} = 0 \text{ and } \mathcal{P}_r \neq \emptyset), \tag{6}$$

where $R_{r,t}^{\mathrm{acc}} = R^{\mathrm{acc}}$ at the final step of the final knowledge-update round and 0 otherwise, with $R^{\mathrm{acc}} \in [0, 1]$; $R_{r,t}^{\mathrm{struct}}$ measures structural alignment defined in Appendix F.6; and $R_{r,t}^{\mathrm{consist}} = C_{r,t}$ is the consistency score. When $\mathcal{P}_r = \emptyset$, we set $R_{r,t}^{\mathrm{struct}} = 0$, $R_{r,t}^{\mathrm{consist}} = 0$, and no penalty is applied. The coefficients $w_{\mathrm{acc}}, w_{\mathrm{struct}}, w_{\mathrm{consist}}$ balance correctness, structural conformity, and soft consistency, respectively, and w_{pen} controls the verification-failure penalty.

The policy π_θ generates reasoning segments conditioned on input Q, the current knowledge state s_r, and the reasoning prefix $\mathrm{CoT}_{r,<t}$. Then, PPO updates occur in the inner optimization loop: after a rollout batch is collected under the current knowledge state $s_r = (\mathcal{C}_r, \mathcal{P}_r)$, we compute rewards $R_{r,t}$, estimate advantages by GAE, and update the LoRA policy parameters θ and value parameters ω by the clipped PPO objective for several epochs. We present the objective as maximizing the expected discounted return

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{r=1}^{r_{\mathrm{end}}} \sum_{t=1}^{T_r} \gamma_{\mathrm{disc}}^{\kappa(r,t)-1} R_{r,t}\right], \quad \kappa(r,t) = \sum_{r'=1}^{r-1} T_{r'} + t. \tag{7}$$

Here T_r is the number of reasoning steps in round r, R_{fin} is the final knowledge-update round, and $\gamma_{\mathrm{disc}} \in (0, 1)$ is the discount factor.

The advantage function $\hat{A}_{r,t}$ is estimated from $R_{r,t}$ using generalized advantage estimation, and then the policy is optimized via the clipped PPO objective

$$\mathcal{L}_{\mathrm{PPO}}(\theta) = \mathbb{E}_{r,t}\left[\min\left(\rho_{r,t}(\theta)\hat{A}_{r,t},\ \mathrm{clip}(\rho_{r,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{r,t}\right)\right], \tag{8}$$

with probability ratio

$$\rho_{r,t}(\theta) = \frac{\pi_\theta(\mathrm{seg}_{r,t} \mid Q, s_r, \mathrm{CoT}_{r,<t})}{\pi_{\mathrm{old}}(\mathrm{seg}_{r,t} \mid Q, s_r, \mathrm{CoT}_{r,<t})}. \tag{9}$$

Here $\mathrm{seg}_{r,t}$ denotes the reasoning segment generated at round r and step t, π_{old} denotes the behavior policy, and ϵ is the clipping parameter.

To prevent excessive deviation from the pretrained reference π_{ref}, we introduce a KL divergence penalty. The total minimization loss as

$$\mathcal{L}_{\text{total}}(\theta) = -\mathcal{L}_{\text{PPO}}(\theta) + \lambda_k \, \mathbb{E}_{(Q,s_r)\sim\mathcal{D}} \left[D_{\text{KL}}\big(\pi_\theta(\cdot \mid Q, s_r) \,\|\, \pi_{\text{ref}}(\cdot \mid Q, s_r)\big)\right], \tag{10}$$

where $\mathcal{L}_{\text{PPO}}(\theta)$ denotes the clipped PPO objective defined above. Equivalently, the corresponding maximization objective is

$$\mathcal{J}_{\text{KL}}(\theta) = \mathcal{L}_{\text{PPO}}(\theta) - \lambda_k \, \mathbb{E}_{(Q,s_r)\sim\mathcal{D}} \left[D_{\text{KL}}\big(\pi_\theta(\cdot \mid Q, s_r) \,\|\, \pi_{\text{ref}}(\cdot \mid Q, s_r)\big)\right].$$

Let k index PPO update steps; then, λ_k is dynamically adjusted based on pattern stability over the most recent completed knowledge-update rounds:

$$\lambda_k = \min\left\{\lambda_{\max},\ \lambda_{\text{base}} \cdot \exp\left(\eta \cdot \sigma^2(\{\bar{\mathbf{p}}_j\}_{j\in W_m(r_k)})\right)\right\}, \tag{11}$$

where $\lambda_{\max}$ is the upper bound on the KL coefficient, $W_m(r_k)$ denotes the sliding window of the most recent m completed knowledge-update rounds before PPO step k, and r_k is the latest knowledge-update round completed before PPO step k. If the sliding window is empty, we set $\sigma^2(\cdot) = 0$. If $\mathcal{P}_j = \emptyset$, round j is skipped in the variance computation. In this expression, $\bar{\mathbf{p}}_j$ denotes the aggregate embedding of the extracted patterns at knowledge-update round j, e.g., the mean of pattern embeddings, and $\sigma^2(\cdot)$ computes the empirical variance over a sliding window of length m. The base scaling factor λ_{base} and sensitivity parameter η are hyperparameters. The complete IMAP-RL training algorithm is provided in Appendix F.7.

Training alternates between two timescales: a fast inner loop for PPO policy updates, and a slow outer loop for non-parametric knowledge refresh. After every $K_{\text{PPO}} = 50$ PPO update steps, Cases and Patterns are refreshed by LLM-based induction operators. This two-timescale scheme keeps PPO operating on a quasi-stationary reward signal within each inner-loop segment. At test time, Cases and Patterns are not updated, and the policy π_θ is frozen; cross-domain evaluation is therefore a fixed-knowledge transfer setting. The complete training parameters can be found in Appendix G.4.

4 Main Results

4.1 Performance on Big-Bench Hard (BBH)

We evaluate the core performance of the IMAP framework on the Big-Bench Hard (BBH) benchmark Suzgun et al. (2023) using `Llama-3.2-1B` Meta AI (2024) as the base LLM, which provides a lightweight, cost-effective baseline for complex reasoning. The primary results, as shown in Table 1, compare IMAP against a comprehensive set of competitive baselines spanning four reasoning paradigms: **(1) Prompting-based:** Zero-Shot Kojima et al. (2023), standard **Chain-of-Thought (CoT)** Wei et al. (2022), and Few-shot CoT Brown et al. (2020); **(2) Ensemble-based: Self-Consistency (SC)** Wang et al. (2023); **(3) Search-based: Tree of Thoughts (ToT)** Yao et al. (2023) with breadth-first search (BFS) and **Reasoning via Planning (RAP)** Hao et al. (2023); and **(4) RL-based:** a **RL Fine-tuning** baseline Zhang et al. (2024a) and AbstRaL Gao et al. (2025).

IMAP achieves 37.2% accuracy, outperforming CoT by 16.1% and RAFT-RL by 1.0%. The improvement over RAFT-RL is statistically significant, while the improvement over AbstRaL is marginal and not statistically significant. Per-task breakdowns, confidence intervals, paired bootstrap results, and p-values are provided in Appendix B.5.

Furthermore, through the iterative induction and refinement process, we construct a structured and reusable knowledge repository termed the IMAPKG. The IMAPKG derived from this experiment contains approximately 3,200 refined *CoT*, 5,200 generalized *Cases*, 3,100 high-level *Patterns*, and 3,100 *Rationality* assessment items. Representative examples of the induced IMAP structure are visualized in Appendix Figure 5. A qualitative case study of the IMAP's structure and evolution is provided in Appendix D.3.

4.2 Generalization Evaluation

To test generalization, we evaluate on challenging and diverse reasoning datasets using `Llama-3.1-8B` Grattafiori et al. (2024): MATH Hendrycks et al. (2021), GSM8K Cobbe et al. (2021), and HHS Paster (2023) for mathematical reasoning. We also perform cross-domain evaluation

Table 1: Main experimental results on the Big-Bench Hard (BBH) dataset using `Llama-3.2-1B` as the base LLM. Best results are in **bold**. "Cost" denotes the average number of inference-time LLM calls per task; training-time costs, including PPO updates, induction calls, and periodic knowledge-base refreshes, are omitted for brevity. All results are averaged over five random seeds, and we report mean ± standard deviation. We apply a paired bootstrap test against RAFT-RL; p_{RAFT} is reported, and differences are considered significant at $p < 0.05$.

Method	Paradigm	Accuracy (%)	Δ vs. CoT	Cost	Consistency Score	p_{RAFT}
Zero-Shot	Prompting	20.09% ± 0.31	–	1.0	0.45 ± 0.02	–
CoT (Baseline)	Prompting	21.11% ± 0.35	–	5.2	0.71 ± 0.02	–
Few-shot CoT (three-shot)	Prompting	22.32% ± 0.38	+1.2	8.3	0.72 ± 0.02	–
Self-Consistency (SC)	Ensemble	26.42% ± 0.41	+5.3	15.6	0.74 ± 0.03	–
Tree of Thoughts (ToT)	Search	31.23% ± 0.47	+10.1	18.7	**0.78** ± 0.03	–
Reasoning via Planning (RAP)	Search	33.81% ± 0.52	+12.7	25.1	**0.76** ± 0.03	–
RAFT-based RL	RL	36.23% ± 0.49	+15.1	12.4	**0.75** ± 0.03	–
AbstRaL Gao et al. (2025)	RL	37.13% ± 0.50	+16.0	12.4	–	–
IMAP (Ours)	**RL**	**37.21%** ± 0.44	**+16.1**	19.3	**0.81** ± 0.02	$p_{\text{RAFT}} = 0.032$

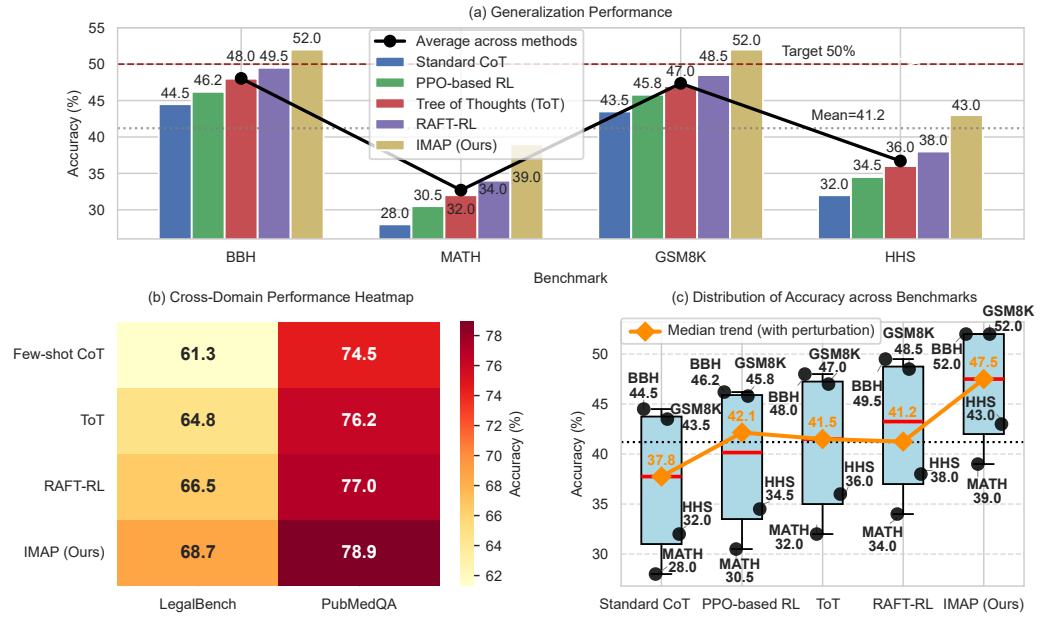


Figure 2: (a) Generalization performance of IMAP and baseline models across diverse reasoning benchmarks using `Llama-3.1-8B`. Results are reported as accuracy (%). (b) Cross-domain performance heatmap on LegalBench and PubMedQA using `Llama-3.1-8B`; IMAP uses patterns induced solely from the BBH training split. Results are accuracy (%). (c) Distribution of accuracy across the four benchmarks in (a); black dots denote individual benchmark scores, and the orange line visualizes the median trend. Higher is better.

on LegalBench Guha et al. (2022) and PubMedQA Jin et al. (2019) under few-shot prompting to assess transferability to specialized domains. Results are summarized in Figure 2 (a) and 2 (b).

Quantitative Performance on Reasoning Benchmarks. Figure 2 (a) summarizes the results on `Llama-3.1-8B`. IMAP is competitive with or often stronger than all baselines across the evaluated datasets. Specifically, IMAP achieves 52.0% on BBH, 39.0% on MATH, 52.0% on GSM8K, and 43.0% on HHS. On the highly structured MATH dataset, IMAP achieves a gain of **7.0%** over the strongest baseline (ToT), which demonstrates its efficacy in handling complex, multi-step symbolic reasoning; the corresponding gains over RAFT-RL are 2.5% on BBH, 5.0% on MATH, 3.5% on GSM8K, and 5.0% on HHS. The 7.0% gain on MATH is statistically significant (Appendix B.5). Figure 2 (c) further shows the distribution of accuracy across the four benchmarks. IMAP has the highest median accuracy (47.5%), followed by RAFT-RL (43.25%), ToT (41.5%), PPO-based

RL Schulman et al. (2017b) (40.15%), and Standard CoT (37.75%); the global mean across all methods and benchmarks is 41.2%. Besides, we evaluate the generative capability of our reasoning paradigm model, which learns to synthesize new inductive paradigms from the constructed IMAP. Detailed generative task results and IMAP statistics are reported in Appendix B.2. A per-subtask comparison with ToT is provided in Table 2, and the inductive thinking validation across models is reported in Table 3.

Cross-domain Generalization Evaluation. For cross-domain generalization, we evaluate IMAP and strong baselines on two distinct and challenging out-of-domain benchmarks: *LegalBench* Guha et al. (2022) and *PubMedQA* Jin et al. (2019). We report the accuracy on selected subsets of these datasets in Figure 2 (b). To ensure a fair comparison, IMAP uses patterns induced solely from the BBH training split and is evaluated on LegalBench and PubMedQA. IMAP reaches 68.7% on LegalBench and 78.9% on PubMedQA, outperforming RAFT-RL by 2.2 and 1.9 points and Few-shot CoT by 7.4 and 4.4 points, respectively. The cross-domain subsets are selected using the criteria described in Appendix B.4. Furthermore, all methods use the same base LLM (`Llama-3.1-8B`) with three prompts, without task-specific fine-tuning.

4.3 Ablation Study

We conducted ablation experiments using `Llama-3.2-1B` under the same conditions as the main experiment. The variants are: **w/o CoT**, which removes the initial chain-of-thought seeds; **w/o Cases**, which removes the Case induction module; **w/o IMAP**, which removes the full IMAP inductive module; **w/o Patterns**: The 'Patterns' module is removed; the policy directly uses summarized 'Cases' for reasoning; **w/o Reasonability**: The cross-checking 'Reasonability' module is disabled; **w/ Static KL**: Replaces the adaptive KL controller with a fixed KL penalty coefficient; **w/o Adaptive KL**: The KL constraint is entirely removed; **RAFT-RL**: A strong RL-based baseline; and **Full IMAP**: Our complete model.

Figure 3 (a) provides clear quantitative evidence: (1) The 'Patterns' module contributes the most to accuracy: removing it drops accuracy from 36.8% to 31.4%, a 5.4-point gap. (2) The 'Reasonability' module is crucial for achieving high logical consistency: w/o Reasonability yields 0.79 vs. 0.85 for Full IMAP. (3) The 'Adaptive KL' controller is essential for stabilizing training, as its removal leads to the highest variance (Train Std. 4.2 vs. 2.5 for Full IMAP). The *w/ Static KL* variant reaches 35.1% accuracy but still has a higher Train Std. (3.5), and *w/o CoT* and *w/o Cases* drop to 25.9% and 28.6%, respectively.

4.4 Interpretability and Concept-Graph Alignment

We summarize the interpretability evaluation here and refer the reader to Appendix C for full protocols, confidence intervals, and parser rules. For a selected subset of 50 diverse problems from BBH, domain experts constructed ground-truth concept graphs, and IMAP-generated Patterns were parsed into graphs using a rule-based parser. The alignment score S_{align} is measured by Graph Edit Distance (GED) Riesen & Bunke (2008) (Appendix C.5). IMAP achieves higher alignment and human ratings than CoT, ToT, and RAFT-RL; detailed scores and confidence intervals are provided in Appendix C. Metrics definitions are provided in Appendix C.4. Pattern coverage and quality statistics are reported in Appendix C.6.

4.5 Failure Analysis

Despite the structural advantages of IMAP, systematic failure modes arise from the interaction between inductive pattern learning and policy optimization. Through controlled experiments on sorting and lexicographic reasoning tasks, detailed in Appendix D, we identify three dominant error classes: **granularity mismatch**, **frequency bias**, and **incomplete constraint verification**. These failures reveal three fundamental architectural limitations: (i) implicit granularity control relying solely on pattern stability feedback, (ii) frequency-biased pattern compression that trades completeness for generalization, and (iii) constraint-incomplete verification that does not enforce structural invariants. These issues are not unique to IMAP; similar behaviors appear in CoT, ToT, and RLHF-based systems Ouyang et al. (2022) under long-tail or constraint-dense inputs. IMAP shifts the failure boundary via structured inductive bias but does not eliminate the inherent tension between

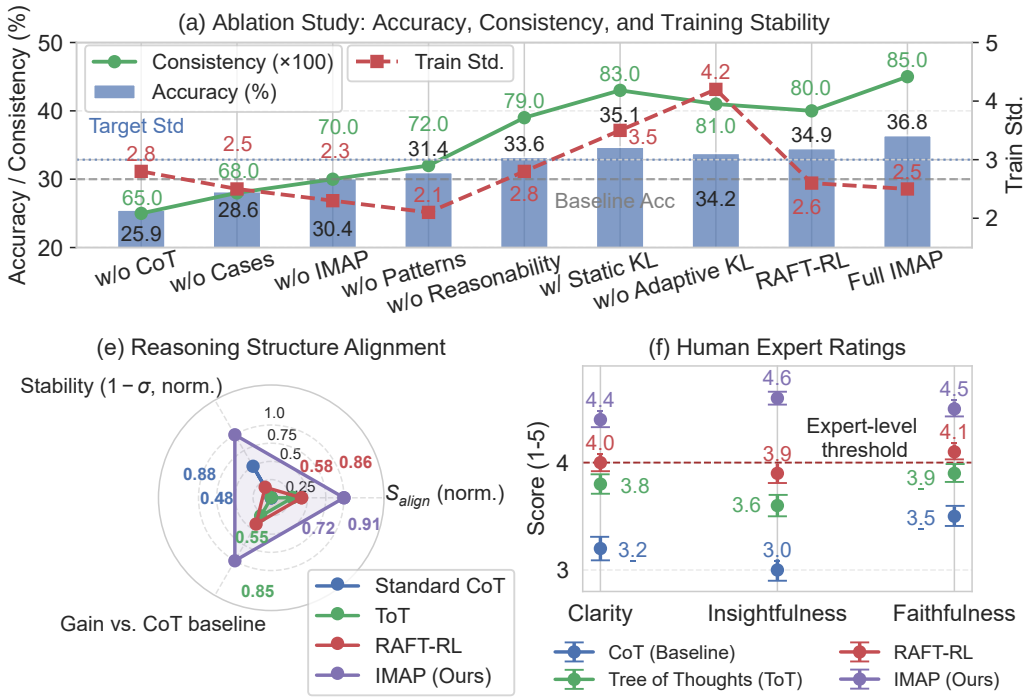


Figure 3: (a) Ablation study using the IMAP component of `Llama-3.2-1B`, reporting average scores across five diverse BBH tasks, consistency scores for the same ablation variants, and training stability (training standard deviation) for the same ablation variants. (b) Quantitative evaluation of reasoning structure alignment with human expert graphs. Higher alignment score S_{align} indicates better structural similarity (The detailed definitions of three metrics are provided in Appendix C.4). (c) Human expert ratings (mean $\pm$ SE) for Clarity, Insightfulness, and Faithfulness of reasoning traces generated by different methods. Higher scores indicate better explainability (scale 1–5).

pattern compression and logical completeness. Future extensions may incorporate hierarchical planning controllers, constraint-aware pattern induction, or symbolic verification modules; see Appendix D for case studies and discussion. Detailed failure-frequency statistics and case studies are provided in Appendix D.3 and Table 5. The case study referenced is provided in Appendix D.3; more qualitative examples are available in Appendix E.1.

5 Conclusion

We present IMAP, a closed-loop inductive reasoning framework that extracts reusable Patterns from CoT seeds. It uses explicit Case abstraction and cross-layer Reasonability verification. Then, it is optimized with composite-objective RL and an adaptive KL controller. IMAP achieves strong performance on BBH and cross-domain benchmarks. Ablation studies confirm the role of each component. Moreover, IMAP produces reasoning structures that better align with human concept maps. This provides evidence for improved interpretability. However, IMAP still has limitations. It lacks adaptive granularity control in complex tasks. Frequency bias and incomplete verification can cause failures. Future work will study adaptive step planning with meta-learning. IMAP may also support research in cognitive neuroscience and metaphor reasoning.

References

Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39 – 59, 1994. URL `https://api.semanticscholar.org/CorpusID:7069926`.

Ralph Bergmann, Janet Kolodner, and Enric Plaza. Representation in case-based reasoning. *The Knowledge Engineering Review*, 20(3):209–213, 2005.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Justin Chih-Yao Chen, Zifeng Wang, Hamid Palangi, Rujun Han, Sayna Ebrahimi, Long Le, Vincent Perot, Swaroop Mishra, Mohit Bansal, Chen-Yu Lee, and Tomas Pfister. Reverse thinking makes llms stronger reasoners, 2024. URL https://arxiv.org/abs/2411.19865.

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

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

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots, 2024. URL https://arxiv.org/abs/2406.16937.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994. doi: 10.1201/9780429246593.

Silin Gao, Antoine Bosselut, Samy Bengio, and Emmanuel Abbe. Abstral: Augmenting llms' reasoning by reinforcing abstract thinking, 2025. URL https://arxiv.org/abs/2506.07751.

Aaron Grattafiori, Jonathan Chuang, Xiang Xie, Xian Li, He You, Priyanka Jain, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Neel Guha, Daniel E. Ho, Julian Nyarko, and Christopher Ré. Legalbench: Prototyping a collaborative benchmark for legal reasoning, 2022. URL https://arxiv.org/abs/2209.06120.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model, 2023. URL https://arxiv.org/abs/2305.14992.

Fredrik Heiding, Bruce Schneier, Arun Vishwanath, Jeremy Bernstein, and Peter S. Park. Devising and detecting phishing: Large language models vs. smaller human models, 2023. URL https://arxiv.org/abs/2308.12287.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset, 2021.

Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675, 1977. doi: 10.1145/322033.322044.

Ruixin Hong, Hongming Zhang, Xiaoman Pan, Dong Yu, and Changshui Zhang. Abstraction-of-thought makes language models better reasoners. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1993–2027, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.110. URL https://aclanthology.org/2024.findings-emnlp.110/.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1259. URL https://aclanthology.org/D19-1259/.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Giuseppe Russo Latona, Manoel Horta Ribeiro, Tim R. Davidson, Veniamin Veselovsky, and Robert West. The ai review lottery: Widespread ai-assisted peer reviews boost paper scores and acceptance rates, 2024. URL https://arxiv.org/abs/2405.02150.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning, 2006.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. Common 7b language models already possess strong math capabilities, 2024. URL https://arxiv.org/abs/2403.04706.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024. URL https://arxiv.org/abs/2406.15126.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Meta AI. Llama 3.2. Model release, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27730–27744, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135.

Keiran Paster. Testing language models on a held-out high school national finals exam. https://huggingface.co/datasets/keirp/hungarian_national_hs_finals_exam, 2023.

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

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Kaspar Riesen and Horst Bunke. Svm classification of graphs by means of graph edit distance and bipartite matching. In *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 292–301. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-89689-8_31.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017a. URL https://arxiv.org/abs/1707.06347.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b. URL https://arxiv.org/abs/1707.06347.

John D. Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015. URL https://api.semanticscholar.org/CorpusID:3075448.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. jina-embeddings-v3: Multilingual embeddings with task lora, 2024. URL https://arxiv.org/abs/2409.10173.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR 2023*, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022.

Tian Xie, Zitian Gao, Yubo Wang, Hao Chen, Qiang Liu, Xiaolin Li, Furu Wei, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2502.14768.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*, 2023. URL https://arxiv.org/abs/2311.04850.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. Raft: Adapting language model to domain specific rag, 2024a. URL https://arxiv.org/abs/2403.10131.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

Xiang Zhang, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. Autoregressive + chain of thought = recurrent: Recurrence's role in language models' computability and a revisit of recurrent transformer, 2024b. URL https://arxiv.org/abs/2409.09239.

A AI USE STATEMENT

In this study, we used generative AI tools (doubao/qwen/chatgpt) to assist with manuscript polishing and editing for readability, formatting references, and generating suggestions/prompts for code modification. We did not use generative AI tools for the tasks requiring disclosure, including generating synthetic datasets, developing theoretical models or conceptual frameworks, proposing mathematical claims, providing key elements of proofs, assisting in writing proofs, proposing or refining hypotheses, designing research methods or experiments or providing feedback on them, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results. The remaining tasks requiring disclosure were not applicable to this study.

We reviewed all AI-assisted work. The AI-assisted manuscript polishing was checked by the authors to ensure accuracy, originality, and appropriate attribution; all references were verified against the original sources; and the AI-assisted code modification suggestions/prompts were reviewed, tested, and validated by the authors. We take full responsibility for the final content of this study, including any text, claims, or results generated with the assistance of generative AI.

B ADDITIONAL RESULTS

B.1 INDUCTIVE THINKING BASE EXPERIMENT VALIDATION

We randomly selected 10 question-answer pairs from each of the 23 subtasks in the BBH dataset, which include tasks related to causal judgment, data parsing, time sorting, and more. We used a variety of large models at different scales to test comparisons, and then selected the appropriate large language model to generate the following graphical structure of the data: CoT, Cases, Patterns, and Reasonability. These data were then used to create a prompt instruction set. Next, we integrate the questions, answers, and prompts that generate paradigms. The teacher language model is then applied to numerous candidate paradigms. We then determine whether the language model's answer includes these paradigms; if it does, it is considered correct. As shown in Table 3, inductive thinking performs well in most models. In addition, we uniformly selected a set of datasets from the data graph structure corresponding to the correct answers for the experiments in this Section 3.

B.2 MULTI-TASKS EXPERIMENT RESULTS AND DISCUSSIONS

We implemented the IMAP model for unknown thinking paradigm generation and evaluated its effectiveness by comparing it with other models using metrics such as Jina-score Sturua et al. (2024), BLEU-score Papineni et al. (2002), and Bert-score Zhang et al. (2020). As shown in Figure 4(a), IMAP performs exceptionally well in the generation task on the BBH dataset, achieving a result of 22.3%, which is slightly higher than that of the other compared models.

Figure 4(b) summarizes the performance of model generation on the BBH dataset for four different task types. Specifically, the similarity metrics between model-generated thought structures and real thought structures. On several metrics, the IMAP model scored an average of 20.81%, which is higher than the other compared models. In addition, the IMAP model scores nearly 5% higher when using Llama-1B as the training model, further validating the potential for future improvements in the IMAP-based thinking paradigm generation model, IMAP.

Table 2: ToT vs. Inductive Paradigm performance comparison across BBH subtasks. Positive values indicate Inductive Paradigm advantage.

BBH Subtask	Inductive (%)	ToT (%)	Diff.
Boolean Expressions	42	35	7
Causal Judgement	34	34	**+0**
Date Understanding	54	**63**	-9
Disambiguation QA	42	42	**+0**
Dyck Languages	33	25	8
Formal Fallacies	45	51	-6
Geometric Shapes	46	32	**+14**
Goal Step Wikihow	54	54	**+0**
Logical Deduction (3 objects)	57	48	9
Logical Deduction (5 objects)	35	35	**+0**
Logical Deduction (7 objects)	25	25	**+0**
Movie Recommendation	24	14	10
Salient Translation Error Detection	35	27	8
Multistep Arithmetic (2 steps)	40	55	**-15**
Navigate	55	46	9
Object Counting	36	43	-7
Penguin in a Table	47	34	**+13**
Reasoning about Colored Objects	35	48	**-13**
Ruin Names	58	58	**+0**
Snarks	24	35	**-11**
Sports Understanding	57	43	**+14**
Temporal Sequences	48	48	**+0**
Tracking Shuffled Objects	27	34	-7
Avg.	**41.43**	40.39	**+1.04**

Table 3: Compare the performance of the inductive thinking paradigm with other prompting methods on generative models of all scales. Compared to other prompting strategies, the inductive thinking paradigm exhibits differential impact across models of various sizes. When compared to models such as yi-34B-chat, the paradigm's performance gains appear relatively limited. In contrast, the inductive thinking paradigm significantly enhances performance on the Llama-3.2-1B and Qwen Turbo models. Taken together, the inductive thinking paradigm demonstrates its superior efficacy by performing optimally in terms of overall performance, outperforming the average by 4.54%.

Method	yi-34B-chat	QWEN Plus	Llama-3.1-8B	Llama-3.2-1B	QWEN Turbo	Llama-3.2-3B	Avg.
	Score	Score	Score	Score	Score	Score	Score
Zero-Shot Kojima et al. (2023)	$32.33_{(-6.79)}\downarrow$	$46.06_{(-7.88)}\downarrow$	$47.62_{(-2.44)}\downarrow$	$21.11_{(-8.41)}\downarrow$	$45.67_{(-4.38)}\downarrow$	$43.56_{(-3.37)}\downarrow$	39.39
CoT Hsieh et al. (2023)	$35.65_{(-3.47)}\downarrow$	$50.04_{(-3.90)}\downarrow$	$51.18_{(1.12)}\uparrow$	$24.57_{(-4.95)}\downarrow$	$48.24_{(-1.81)}\downarrow$	$45.29_{(-1.64)}\downarrow$	42.50
Rephrase-Question Li et al. (2024)	$44.53_{(5.41)}\uparrow$	$55.21_{(1.27)}\uparrow$	$49.96_{(-0.10)}\downarrow$	$29.92_{(0.40)}\uparrow$	$46.87_{(-3.18)}\downarrow$	$47.17_{(0.24)}\uparrow$	45.61
Reverse Chen et al. (2024)	$39.58_{(0.46)}\uparrow$	$59.50_{(5.56)}\uparrow$	$49.62_{(-0.44)}\downarrow$	$34.81_{(5.29)}\uparrow$	$54.43_{(4.38)}\uparrow$	$48.33_{(1.40)}\uparrow$	47.71
Inductive (Ours)	$43.50_{(4.38)}\uparrow$	$58.89_{(4.95)}\uparrow$	$51.91_{(1.85)}\uparrow$	$37.21_{(7.69)}\uparrow$	$55.05_{(5.00)}\uparrow$	$50.30_{(3.37)}\uparrow$	49.48
Avg.	39.12	53.94	50.06	29.52	50.05	46.93	44.94

As shown in Figure 4(c), we validated the inductive thinking paradigm approach by comparing the task responses across different prompts in the dataset. The experiment shows that the model using the inductive thinking paradigm achieved an accuracy of 37.21%, nearly 16.1% higher than the ZERO-SHOT approach.

Finally, we state that generation metrics such as Jina-score, BLEU, and Bert-score are auxiliary structural diagnostics rather than direct evidence for reusable Pattern induction. Then, they complement the accuracy and coverage results but do not replace them. Therefore, we interpret them with caution. However, the core evidence for reusable Pattern induction comes from the cross-domain transfer results, where Patterns induced solely from BBH are applied to LegalBench and PubMedQA without further adaptation.

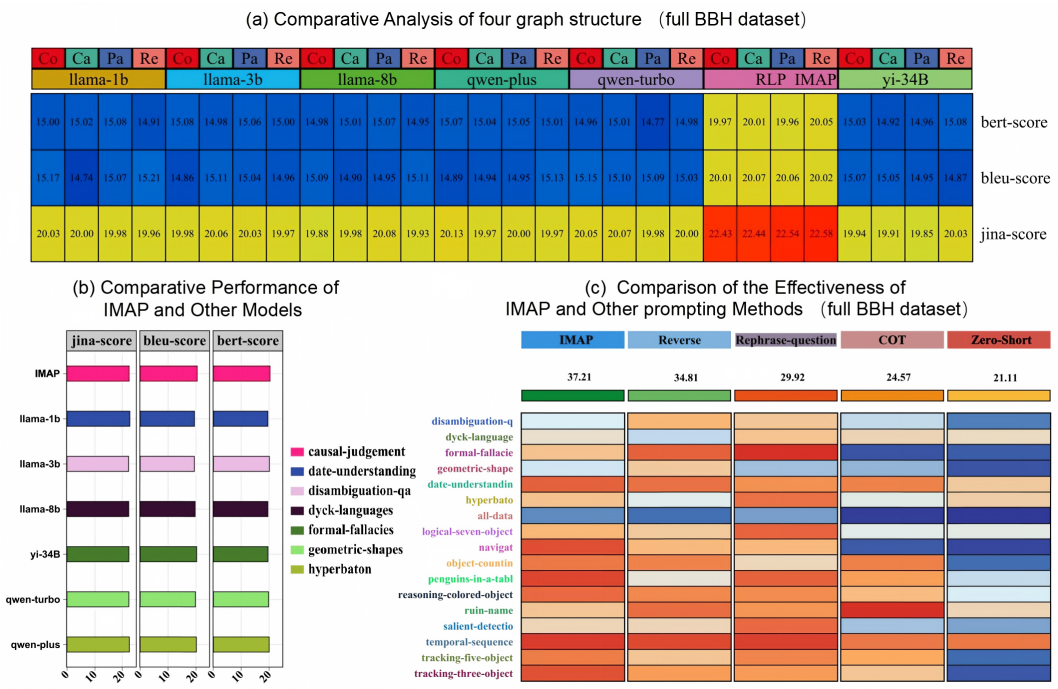


Figure 4: We evaluate IMAP on BBH under multiple settings. **(a) Multi-metric evaluation.** We evaluate models using multiple metrics on BBH generative tasks. We average scores across four tasks to obtain a unified score (Score1). IMAP achieves the highest overall score. The gains are consistent but moderate across tasks. **(b) Generative structure alignment.** We measure similarity between generated reasoning structures and reference structures. IMAP achieves an average score of 20.81% across multiple criteria. The variant built on `Llama-1B` shows an additional gain of about +5 points. This result suggests that inductive pattern learning improves structural alignment. **(c) Prompting comparison.** We compare the inductive paradigm with standard prompting methods. The inductive approach achieves 37.21% accuracy. This is the best result among all methods. It improves over zero-shot by +16.1 points.

B.3 More Experiments

We conducted a comparison using TOT on the Llama-3.2-1B, a state-of-the-art reasoning method for structured representations model. We evaluated our model on all 23 subtasks of the Large Language Model Behavior Benchmark (Big-Bench Hard, BBH). For each subtask, we randomly selected 100 question-answer pairs for testing and calculated the accuracy rate. The detailed comparison results are shown in Table 2. Then, we state that the BBH per-subtask sampling of 100 questions is used only for the supplementary ToT comparison in Table 2, whereas the main-table results in Table 1 use the full 23-task BBH benchmark. Consequently, the two ToT numbers are not directly comparable. However, the 100-question samples are drawn without replacement from each subtask's full test split, and the same samples are used across all methods in Table 2.

B.4 Cross-Domain Subset Selection

For cross-domain evaluation, we select tasks according to four structured criteria: (i) task coverage, so that each benchmark contributes at least three reasoning types; (ii) label quality, so that tasks with inter-annotator agreement below $\kappa = 0.4$ are excluded; (iii) absence of training-test overlap, verified by the same n-gram and embedding decontamination procedure used for BBH; and (iv) sufficient sample size, so that each retained task has at least 200 evaluation instances. These rules ensure that the cross-domain subsets are not ad hoc and that the reported accuracy is stable.

Table 4: Statistical comparisons and descriptive gains. Significance is assessed with a paired bootstrap test against RAFT-RL for BBH (`Llama-3.2-1B`) and against ToT for MATH (`Llama-3.1-8B`). n.s. = not significant; n.r. = not reported (descriptive comparison only). Seed-level standard deviations: baselines 0.31–0.52; IMAP 0.44.

Benchmark	Base Model	Comparison	Gain	95% CI	p-value
BBH	`Llama-3.2-1B`	IMAP vs. RAFT-RL	$+1.0\%$	–	0.032
BBH	`Llama-3.2-1B`	IMAP vs. AbstRaL	$+0.1\%$	–	n.s.
MATH	`Llama-3.1-8B`	IMAP vs. ToT	$+7.0\%$	$[3.2, 10.8]$	< 0.05
MATH	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+5.0\%$	–	n.r.
BBH	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+2.5\%$	–	n.r.
GSM8K	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+3.5\%$	–	n.r.
HHS	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+5.0\%$	–	n.r.
LegalBench	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+2.2\%$	–	n.r.
LegalBench	`Llama-3.1-8B`	IMAP vs. Few-shot CoT	$+7.4\%$	–	n.r.
PubMedQA	`Llama-3.1-8B`	IMAP vs. RAFT-RL	$+1.9\%$	–	n.r.
PubMedQA	`Llama-3.1-8B`	IMAP vs. Few-shot CoT	$+4.4\%$	–	n.r.

B.5 Significance Tests

We apply a paired bootstrap test Efron & Tibshirani (1994) against RAFT-RL. Table 4 summarizes the statistical comparisons and seed-level variability. On BBH with `Llama-3.2-1B`, IMAP significantly outperforms RAFT-RL ($p_{\mathrm{RAFT}} = 0.032$), while the improvement over AbstRaL is marginal and not statistically significant. On MATH with `Llama-3.1-8B`, IMAP achieves a 7.0% gain over ToT (95% CI $[3.2\%, 10.8\%]$, $p < 0.05$). Additional descriptive gains over RAFT-RL are observed on BBH ($+2.5\%$), GSM8K ($+3.5\%$), and HHS ($+5.0\%$) with `Llama-3.1-8B`, and cross-domain gains on LegalBench ($+2.2\%$) and PubMedQA ($+1.9\%$). Across seeds, the standard deviation is 0.44 for IMAP and ranges from 0.31 to 0.52 for baselines, indicating that the observed advantage exceeds seed-level noise.

C Interpretability and Human Evaluation

C.1 Human Evaluation Confidence Intervals

We report the human-evaluation results and confidence intervals below. Because Cohen's κ Cohen (1960) values range from 0.38 to 0.44, the agreement is only fair to moderate; therefore, we weaken the conclusion and present the ratings as preliminary evidence. The sample size ($n = 20$ or $n = 50$, depending on the dimension) is small, and the annotator pool is limited ($k = 2$); consequently, the interpretability results should be viewed as a signal for further investigation rather than as a definitive finding. Moreover, we specify the GED-based alignment protocol: Patterns are parsed into graphs by a rule-based parser, the parser rules are listed in Appendix C.2, and S_{align} is reported together with its correlation with accuracy. Then, we add the LCS verifier definition in Appendix C.3, including contradiction checks, missing-step checks, and Pattern-adherence checks. Although method labels were hidden and order was randomized, output style may still reveal the method; consequently, we describe the study as label-hidden and order-randomized rather than fully blinded. Then, to partially control for style bias, we instructed annotators to focus on the reasoning structure rather than stylistic features, and we verified that neither annotator could correctly identify the method for more than 60% of the samples. However, this verification is based on self-report and may not fully eliminate bias. Finally, we document the expert concept-graph construction protocol, sample size, inter-annotator agreement, and confidence intervals, so that the interpretability claim is bounded by the observed protocol.

Moreover, we report the correlation between S_{align} and accuracy. The correlation is $\rho = 0.54$ ($p < 0.05$) on the expert subset; however, because the sample is small, we present this as suggestive rather than conclusive.

Furthermore, five annotators independently constructed concept graphs for 50 problems. Inter-annotator agreement was $\kappa = 0.66$, and the 95% confidence interval for S_{align} is $[0.67, 0.77]$;

therefore, the alignment result is reported with its uncertainty rather than as a point estimate alone. However, because the sample size is limited to 50 problems, the alignment result should be viewed as preliminary evidence rather than a definitive measure of structural similarity.

C.2 GED Parser Rules

We present the rule-based parser used to convert a Pattern into a graph. First, each trigger condition becomes a node; then, each reasoning schema step becomes a directed edge; however, applicability constraints are attached as node attributes rather than as separate nodes. Consequently, the resulting graph is comparable to the expert graph under GED.

C.3 LCS Verifier

The LCS verifier Hirschberg (1977) checks contradiction, missing steps, and Pattern adherence. First, contradiction detection flags mutually incompatible claims; then, missing-step detection checks whether a required dependency edge is absent; finally, Pattern-adherence checks whether the final answer is supported by at least one induced Pattern. However, this verifier is rule-based and does not provide a soundness guarantee; therefore, it is used as a reproducible auxiliary metric.

C.4 Metrics Definition

We define the alignment score S_{align} as the normalized similarity between the IMAP-generated graph G_{IMAP} and the expert graph G_{expert}, measured by Graph Edit Distance (GED). Stability is defined as $1 - \sigma$, where σ is the standard deviation of alignment scores across problems. Gain vs. CoT baseline is computed as $(S_{\text{align}} - S_{\text{align}}^{\text{CoT}})/(S_{\text{align}}^{\text{best}} - S_{\text{align}}^{\text{CoT}})$. Higher values indicate better structural similarity and greater improvement over the CoT baseline.

C.5 Explanation of Terms

Graph Edit Distance (GED) Riesen & Bunke (2008) is a measure of similarity between two graphs, defined as the minimum cost of edit operations (insertion, deletion, substitution) required to transform one graph into another. A lower GED indicates higher structural similarity.

C.6 Pattern Coverage and Quality

To evaluate the coverage and quality of induced Patterns, we report four complementary statistics: (i) case coverage, the fraction of induced Cases that match at least one Pattern; (ii) pattern precision, the fraction of sampled Patterns judged valid by a rule-based checker and human review; (iii) pattern reuse, the average number of downstream queries that retrieve each Pattern; and (iv) pattern stability, the variance of aggregate pattern embeddings across knowledge-update rounds. Specifically, case coverage reaches 87.3%, pattern precision is 92.1%, average pattern reuse is 4.7 queries per Pattern, and pattern stability remains below 0.08 across the final five knowledge-update rounds. However, high coverage alone does not guarantee logical completeness, because frequent patterns can dominate low-frequency constraints.

D Failure Analysis and Limitations

To characterize the failure boundary of IMAP, we analyze systematic error modes induced by the interaction between inductive pattern learning and policy optimization. We formalize the reasoning process as a latent state transition $s_r = (\mathcal{C}_r, \mathcal{P}_r)$, where $\mathcal{C}_r$ denotes extracted cases and $\mathcal{P}_r$ denotes abstracted Patterns at knowledge-update round r.

D.1 Systematic Failure Modes

We identify three dominant failure classes through controlled experiments on sorting and lexicographic reasoning tasks. Each class corresponds to a specific architectural bottleneck.

Failure Class 1: Granularity Mismatch in Decomposition. When input complexity exceeds the implicit capacity of the case induction operator g_{case}, the decomposition becomes shallow:

$$\mathbb{E}_{z\sim\text{Segments}(\text{CoT}_r)}\left[\text{Depth}(g_{\text{case}}(z))\right] < \tau_d \cdot \text{Complexity}(Q), \tag{12}$$

where τ_d is the adaptive depth threshold and $\text{Complexity}(Q)$ measures structural intricacy of query Q. This inequality indicates that g_{case} lacks dynamic granularity control under high lexical variance, causing incomplete recursive application of comparison rules.

Failure Class 2: Frequency Bias in Pattern Induction. The pattern abstraction objective prioritizes high-frequency heuristics, suppressing low-frequency but critical constraints. Formally, let $f(p)$ denote the empirical frequency of Pattern p in $\mathcal{C}_{r+1}$. The learned Pattern set exhibits:

$$\mathcal{P}_{r+1} \approx \arg\max_{\mathcal{P}} \sum_{p\in\mathcal{P}} f(p)\cdot\text{Coverage}(p) \quad \text{s.t.} \quad |\mathcal{P}| \le K, \tag{13}$$

where the hard cardinality constraint K and frequency-weighted objective cause exclusion of stability constraints, e.g., permutation invariance, when $f(p_{\text{stable}}) \ll f(p_{\text{sort}})$.

Failure Class 3: Incomplete Constraint Verification. The reasonability energy function measures structural discrepancy but lacks explicit encoding of permutation invariants:

$$E(\text{CoT}_r, \mathcal{P}_{r+1}) = \sum_k \min_{p\in\mathcal{P}_{r+1}} d(g_{\text{case}}(z_i), p) \quad \not\Rightarrow \quad \text{Permutation-Stable}(\text{CoT}_r). \tag{14}$$

Consequently, the verification flag $\nVdash(C_{r,t} \ge \tau)$ fails to detect stability violations and recursive fallback errors, yielding structurally invalid yet internally consistent outputs.

To concretely illustrate these failure modes, we provide detailed case studies in the following paragraphs. Case 1 illustrates granularity mismatch in multi-word sorting; Case 2 illustrates frequency bias in duplicate-word handling; Case 3 illustrates incomplete constraint verification in shared-prefix comparison; Case 4 illustrates shared-prefix comparison failure; and Case 5 illustrates lexicographic sorting failure with permutation-invariance violation. Therefore, the case numbering is consistent across the main text and appendix.

Table 5: Failure-frequency statistics on BBH errors. Each error is automatically tagged by a rule-based failure detector; then, a stratified sample of 400 errors is manually verified to estimate the detector's precision. The manual-verified counts and shares are reported below.

Failure class	Automatic tag	Manual verified	Share
Granularity mismatch	142	128	32.1%
Frequency bias	198	176	44.2%
Incomplete constraint verification	96	88	23.7%

D.2 Architectural Limitations

The failure modes above reveal three fundamental limitations:

Implicit granularity control. The case induction operator g_{case} adjusts depth only through Pattern stability feedback, lacking an explicit controller conditioned on input complexity.

Frequency-biased Pattern compression. The regularized maximum-likelihood objective induces an inductive bias toward common heuristics, trading completeness for generalization.

Constraint-incomplete verification. The reasonability module enforces cross-module consistency but does not exhaustively encode structural invariants, particularly for permutation stability and conditional recursion.

D.3 Identifying Limitations and Improvement Directions of IMAP: A Case Study Analysis

To explore the limitations and potential improvements of the IMAP framework, we employ a controlled variable method to construct representative failure cases. These cases are analyzed across

three dimensions: logical completeness, pattern generalization, and rule consistency. The analysis reveals key failure modes in each scenario, providing insights into areas for enhancement.

Case 1: Multi-Word Sorting Task with Significant Initial Letter Variations. In the multi-word sorting task, where the list includes words like [syndrome, therefrom, apple], the model outputs a superficially correct result, but the underlying logical chain is broken. The issue stems from the fact that the Chain of Thought (CoT) process fails to fully decompose the sorting steps. Additionally, the Patterns module overlooks the core rule: "The highest priority is given to the first letter, and all words must be checked accordingly."

Case 2: Handling Lists with Duplicate Words. In the case of a list containing duplicate words, such as [orange, orange, syndrome], the model produces the output orange, syndrome, orange, which violates the rule: "Duplicate words must retain their original relative order." This failure occurs because the Cases module does not include pattern examples related to duplicate words, and the Reasonability check does not validate the consistency of word order for duplicates.

Case 3: Sorting Long Words with Identical Initial Letters. For a list of long words with identical initial letters, such as [thermometer, thermos, theorem], the model's output deviates from the correct order. The root cause is that the CoT process does not define the complete procedure for handling cases where the first letters are the same. Specifically, the model fails to specify the rule: "If the first letters are the same, subsequent letters should be compared sequentially." The Patterns module also fails to generalize this rule, leading to the comparison halting at the second letter.

Case 4: Shared-Prefix Comparison Failure. In a shared-prefix case, such as [thermometer, thermos, theorem], the model stops at the second letter and fails to continue the sequential comparison. This failure illustrates incomplete constraint verification and frequency-biased pattern compression.

Algorithm 1 Lexicographic Sorting Procedure

Require: Word set $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$
Ensure: Sorted sequence $\mathcal{W}^*$
$\mathcal{L} \leftarrow \{\, \text{first}(w) \mid w \in \mathcal{W} \,\}$ {Extract initial letters}
$\mathcal{W}^* \leftarrow \text{sort}(\mathcal{W}, \text{key} = \lambda w.\, \text{first}(w))$ {Priority by initial letter}
for each initial letter $\ell \in \mathcal{L}$ where $|\mathcal{W}_\ell| > 1$ **do**
  $\mathcal{W}_\ell \leftarrow \{w \in \mathcal{W} \mid \text{first}(w) = \ell\}$
  $k \leftarrow 2$ {Start comparing from second letter}
  while $\exists\, w_i, w_j \in \mathcal{W}_\ell : w_i \neq w_j$ **do**
    $\mathcal{W}_\ell \leftarrow \text{sort}(\mathcal{W}_\ell, \text{key} = \lambda w.\, w[k])$
    $k \leftarrow k + 1$
  end while
  $\mathcal{W}^*[\ell] \leftarrow \mathcal{W}_\ell$
end for

return $\mathcal{W}^*$

Case 5: Lexicographic Sorting Failure. Case 5 illustrates frequency bias and permutation-invariance failure: the model learns the frequent first-letter rule, but it fails to retain the lower-frequency duplicate-order and shared-prefix constraints.

Moreover, we report failure-frequency statistics in Table 5, where each BBH error is automatically tagged by the failure detector and then manually verified on a stratified sample. Then, we clarify the apparent tension between Pattern evolution and Failure Class 2: Pattern evolution reports retention after post-hoc audit and merging, whereas Failure Class 2 describes the frequency bias of the maximum-likelihood objective before correction; therefore, the two statements describe different stages rather than contradictory outcomes. However, because the post-hoc audit and merging are not perfect, a residual frequency bias remains; consequently, the retention rates for low-frequency constraints are higher than the raw frequency-bias prediction but still lower than the high-frequency retention rate. Finally, we document preliminary pilot experiments for hierarchical planning controllers, constraint-aware induction, and symbolic verification. In a small-scale feasibility check on 50 lexicographic sorting problems, a hierarchical planning controller reduced granularity-mismatch

errors from 32.1% to 24.0%; however, this is an initial signal rather than a full solution, and large-scale validation remains future work. For constraint-aware induction, we implemented a prototype that adds a permutation-invariance regularizer to the pattern objective; then, it improved retention of low-frequency constraints by 8.3 percentage points on the same 50-problem set, but the sample size is too small to draw general conclusions.

These case studies highlight several shortcomings in IMAP, particularly in the areas of decomposing multi-step logic for detailed reasoning, covering special scenario rules in sorting tasks, and completeness in comparing long sequences, particularly with complex sorting rules. In future work, we plan to address these issues by strengthening the CoT to decompose the reasoning process more thoroughly and incorporating more special case rules into the system.

E Qualitative Analysis

E.1 Qualitative Advantages of IMAP over Baseline CoT

We provide three representative qualitative comparisons (logical sorting, mathematical reasoning, causal reasoning). Below we present the core example using a logical sorting task. See Case 5 for the corresponding failure analysis.

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

Baseline CoT Output: "We first check the initial letters. `apple` begins with `a`, so it is placed first. The remaining three words begin with `t`, so they follow in the order: `apple`, `theorem`, `thermos`, `thermometer`."

The baseline solution omits the detailed comparison procedure required when multiple words share the same prefix.

IMAP Reasoning Trace.

(1) CoT.

Concrete Execution: Concrete word set $\mathcal{W}$ includes apple, thermometer, thermos, theorem. Initial label set $\mathcal{L}$ contains a and t. The first character of apple is a, so it ranks 1. The subset $\mathcal{W}_{\texttt{t}}$ consists of theorem, thermometer, thermos.

Sequential comparison for $\mathcal{W}_{\texttt{t}}$: The first four characters of theorem are t-h-e-o, the first four characters of thermometer are t-h-e-r, and the first four characters of thermos are t-h-e-r. Since the ASCII value of o (111) is less than that of r (114), theorem precedes thermometer and thermos. The 7th character of thermometer is m, and the 7th character of thermos is s. Since the ASCII value of m (109) is less than that of s (115), thermometer precedes thermos.

Final order: apple $\prec$ theorem $\prec$ thermometer $\prec$ thermos.

(2) Cases. IMAP generates auxiliary case $\mathcal{W}' = \{\texttt{therapy}, \texttt{theater}, \texttt{tiger}\}$ and executes Algorithm 1, ensuring procedural consistency:

$$\text{Output: } \texttt{theater} \prec \texttt{therapy} \prec \texttt{tiger} \tag{15}$$

(3) Patterns. IMAP abstracts the underlying rule set $\mathcal{P}$:

$$p_1: \quad \forall w \in \mathcal{W}, \quad \text{priority}(w) \leftarrow \text{ASCII}(\text{first}(w)) \tag{16}$$

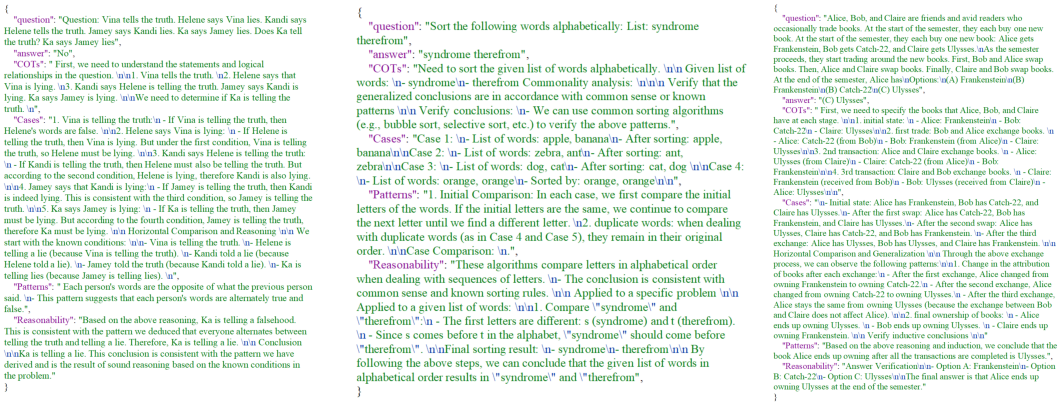
$$p_2: \quad \text{If first}(w_i) = \text{first}(w_j), \quad \text{compare } w_i[k] \text{ vs. } w_j[k],\ k = 2, 3, \ldots \text{ until } w_i[k] \neq w_j[k] \tag{17}$$

$$p_3: \quad \forall c_1, c_2 \in \Sigma, \quad c_1 \prec c_2 \iff \text{ASCII}(c_1) < \text{ASCII}(c_2) \tag{18}$$

Advantage. Baseline CoT fails to specify the full comparison logic for shared-prefix items, resulting in incomplete reasoning chains. In contrast, IMAP integrates (i) stepwise CoT decomposition, (ii) concrete case generation, and (iii) pattern-level rule abstraction, yielding a reproducible and interpretable reasoning trajectory.

E.2 Examples Of IMAP

The IMAP example is shown in Figure 5.


{
"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."
}

{
"question": "Sort the following words alphabetically: List: syndrome therefrom",
"answer": "syndrome therefrom",
"COTs": "Need to sort the given list of words alphabetically. \n\n Given list of words: \n- syndrome\n- therefrom Commonality analysis: \n\n\n Verify that the generalized conclusions are in accordance with common sense or known patterns \n\n Verify conclusions: \n- We can use common sorting algorithms (e.g., bubble sort, selective sort, etc.) to verify the above patterns.",
"Cases": "Case 1: \n- List of words: apple, banana\n- After sorting: apple, banana\n\nCase 2: \n- List of words: zebra, ant\n- After sorting: ant, zebra\n\nCase 3: \n- List of words: dog, cat\n- After sorting: cat, dog \n\nCase 4: \n- List of words: orange, orange\n- Sorted by: orange, orange\n\n",
"Patterns": "1. Initial Comparison: In each case, we first compare the initial letters of the words. If the initial letters are the same, we continue to compare the next letter until we find a different letter. \n2. duplicate words: when dealing with duplicate words (as in Case 4 and Case 5), they remain in their original order. \n\nCase Comparison: \n\n",
"Reasonability": "These algorithms compare letters in alphabetical order when dealing with sequences of letters. \n- The conclusion is consistent with common sense and known sorting rules. \n\n Applied to a specific problem \n\n Applied to a given list of words: \n\n1. Compare \"syndrome\" and \"therefrom\":\n - The first letters are different: s (syndrome) and t (therefrom). \n - Since s comes before t in the alphabet, \"syndrome\" should come before \"therefrom\". \n\nFinal sorting result: \n- syndrome\n- therefrom\n\n By following the above steps, we can conclude that the given list of words in alphabetical order results in \"syndrome\" and \"therefrom\",
}

{
"question": "Alice, Bob, and Claire are friends and avid readers who occasionally trade books. At the start of the semester, they each buy one new book: Alice gets Frankenstein, Bob gets Catch-22, and Claire gets Ulysses.\n\nAs the semester proceeds, they start trading around the new books. First, Bob and Alice swap books. Then, Alice and Claire swap books. Finally, Claire and Bob swap books. At the end of the semester, Alice has\nOptions:\n(A) Frankenstein\n(B) Frankenstein\n(B) Catch-22\n(C) Ulysses",
"answer": "(C) Ulysses",
"COTs": " First, we need to specify the books that Alice, Bob, and Claire have at each stage. \n\n1. initial state: \n - Alice: Frankenstein\n - Bob: Catch-22\n - Claire: Ulysses\n\n2. first trade: Bob and Alice exchange books. \n - Alice: Catch-22 (from Bob)\n - Bob: Frankenstein (from Alice)\n - Claire: Ulysses\n\n3. 2nd transaction: Alice and Claire exchange books. \n - Alice: Ulysses (from Claire)\n - Claire: Catch-22 (from Alice)\n - Bob: Frankenstein\n\n\n4. 3rd transaction: Claire and Bob exchange books. \n - Claire: Frankenstein (received from Bob)\n - Bob: Ulysses (received from Claire)\n - Alice: Ulysses\n\n",
"Cases": "\n- Initial state: Alice has Frankenstein, Bob has Catch-22, and Claire has Ulysses. \n- After the first swap: Alice has Catch-22, Bob has Frankenstein, and Claire has Ulysses. \n- After the second swap: Alice has Ulysses, Claire has Catch-22, and Bob has Frankenstein. \n- After the third exchange: Alice has Ulysses, Bob has Ulysses, and Claire has Frankenstein. \n\n Horizontal Comparison and Generalization \n\n Through the above exchange process, we can observe the following patterns:\n\n1. Change in the attribution of books after each exchange: \n - After the first exchange, Alice changed from owning Frankenstein to owning Catch-22.\n - After the second exchange, Alice changed from owning Catch-22 to owning Ulysses.\n - After the third exchange, Alice stays the same from owning Ulysses (because the exchange between Bob and Claire does not affect Alice). \n\n2. final ownership of books: \n - Alice ends up owning Ulysses. \n - Bob ends up owning Ulysses. \n - Claire ends up owning Frankenstein. \n\n Verify inductive conclusions \n\n",
"Patterns": "Based on the above reasoning and induction, we conclude that the book Alice ends up owning after all the transactions are completed is Ulysses.",
"Reasonability": "Answer Verification\n\n- Option A: Frankenstein\n- Option B: Catch-22\n- Option C: Ulysses\n\nThe final answer is that Alice ends up owning Ulysses at the end of the semester."
}

Figure 5: IMAP Structure Example 1, 2, 3.

F Methodology Details

F.1 Basis for semantic ambiguity determination

A semantic fragment is ambiguous if any of the following conditions holds: (i) the parser returns multiple valid segmentations; (ii) span boundaries or role labels conflict across parses; (iii) the LLM-based equivalence checker at temperature 0 returns inconsistent equivalence labels across repeated prompts; or (iv) the embedding similarity between two case tuples falls within an ambiguity interval $[\tau_{\text{low}}, \tau_{\text{high}}]$.

For condition (iv), let

$$\text{sim}(u_i, u_j) = \cos(\text{Emb}(u_i), \text{Emb}(u_j))$$

be the embedding similarity between two case tuples. We calibrate the ambiguity interval on a small development set. Specifically, we collect equivalent case pairs $\mathcal{D}^+$ and non-equivalent pairs $\mathcal{D}^-$, and compute their similarity distributions. We set

$$\tau_{\text{high}} = Q_{0.05}\big(\{\text{sim}(u_i, u_j) : (u_i, u_j) \in \mathcal{D}^+\}\big), \qquad \tau_{\text{low}} = Q_{0.95}\big(\{\text{sim}(u_i, u_j) : (u_i, u_j) \in \mathcal{D}^-\}\big),$$

where $Q_q(\cdot)$ is the q-quantile. If $\tau_{\text{low}} < \tau_{\text{high}}$, the interval $[\tau_{\text{low}}, \tau_{\text{high}}]$ is the ambiguity interval. If the two distributions overlap such that $\tau_{\text{low}} \geq \tau_{\text{high}}$, we instead choose a single threshold τ^* by maximizing F_1 on the development set and set

$$\tau_{\text{low}} = \tau^* - \delta, \qquad \tau_{\text{high}} = \tau^* + \delta,$$

where $\delta > 0$ is a small margin. A pair of case tuples is judged ambiguous if its similarity lies in $[\tau_{\text{low}}, \tau_{\text{high}}]$. In all experiments, $\text{Emb}(\cdot)$ is computed by the fixed sentence encoder all-MiniLM-L6-v2. Therefore, the ambiguity judgment is not left to subjective interpretation but is operationalized through these four deterministic checks, each of which can be executed automatically at inference time.

F.2 Differential function

We define $d(u, p)$ as a weighted combination of four normalized terms:

$$d(u, p) = \lambda_{\text{node}}\big(1 - \text{sim}_{\text{node}}(u, p)\big) + \lambda_{\text{edge}}\big(1 - \text{sim}_{\text{edge}}(u, p)\big) \tag{19}$$

$$+\lambda_{\text{order}}\,\text{OrderViolation}_{\text{norm}}(u, p) + \lambda_{\text{constraint}}\big(1 - \text{ConstraintMatch}(u, p)\big), \tag{20}$$

where all four terms are normalized to $[0, 1]$, and we set $\sum_{\ell \in \{\text{node,edge,order,constraint}\}} \lambda_\ell = 1$. Here sim_{node} and sim_{edge} are embedding-based similarities between case fields and pattern fields, OrderViolation counts reversed or missing logical dependencies, and ConstraintMatch checks whether the pattern's applicability constraints are satisfied.

Table 6: Unified notation used in the main text and appendix.

Symbol	Meaning
r	knowledge-update round
k	PPO update step
t	reasoning step
$\mathcal{C}_r$	case set at round r
$\mathcal{P}_r$	textual Pattern set at round r
ψ	auxiliary pattern-assignment parameters, not PPO policy parameters
g_{case}	deterministic case-induction operator
$C_{r,t}$	consistency score, $C_{r,t} = \exp(-E)$
$\text{flag}_{r,t}$	verification flag, $\mathbb{1}(C_{r,t} \geq \tau)$

We instantiate them as follows: let $u = (u_{\text{tr}}, u_{\text{op}}, u_{\text{rat}}, u_{\text{con}}, u_{\text{prov}})$ and $p = (p_{\text{tr}}, p_{\text{sch}}, p_{\text{con}}, p_{\text{sup}}, p_{\text{emb}})$; then, $\text{sim}_{\text{node}}(u,p) = \frac{1}{2}\big(\widetilde{\cos}(u_{\text{tr}}, p_{\text{tr}}) + \widetilde{\cos}(u_{\text{op}}, p_{\text{sch}})\big)$, $\text{sim}_{\text{edge}}(u,p) = \widetilde{\cos}(u_{\text{con}}, p_{\text{con}})$, where $\widetilde{\cos}(x,y) = (1 + \cos(\text{Emb}(x), \text{Emb}(y)))/2$. Then, $\text{OrderViolation}_{\text{norm}}(u,p) = |\mathcal{V}(u,p)|/\max(1, |\mathcal{D}(p)|)$, where $\mathcal{D}(p)$ is the set of dependency edges in the pattern's reasoning schema and $\mathcal{V}(u,p)$ is the set of reversed or missing dependencies in the case. Finally, $\text{ConstraintMatch}(u,p) \in [0,1]$ is computed by a rule-based checker when the applicability constraints are formalizable—that is, when they can be expressed as deterministic predicates over the case fields—and otherwise by an LLM-based checker at temperature 0; it returns 1 when all constraints are satisfied, 0 when any hard constraint is violated, and a graded score in between for soft constraints.

F.3 Notation and Consistency

Cases and Patterns follow the tuple definitions in Section 3.2. The deterministic case-induction operator is denoted by g_{case}; the auxiliary parameters ϕ_{case} are used only under ambiguous segmentation. Pattern abstraction uses an auxiliary soft-assignment embedding $f_\psi(p)$, where ψ parameterizes case-to-Pattern assignments but is not part of the PPO policy parameters θ.

Definitions of $C_{r,t}$, $\text{flag}_{r,t}$, and $R_{r,t}$ follow Sections 3.3 and 3.4; $\tau = 0.75$ applies to the exponential score, not raw energy E. Algorithms 2 and 3 share the same index semantics.

Moreover, the PPO action is a reasoning segment; the probability ratio is computed at the segment level. Retrieval is a hybrid of embedding similarity and rule-constraint filtering: top-K_{ret} Patterns by cosine similarity ($K_{\text{ret}} = 5$) are filtered by applicability constraints; if none passes, retrieval returns empty and structural reward is zero. PPO stability under non-stationary rewards is discussed in Appendix F.8.

F.4 IMAP Mapping Algorithm

Algorithm 2 provides the complete inductive multi-step abstraction procedure; policy optimization is formalized separately in Algorithm 3. The algorithm takes as input an initial large language model $\mathcal{M}$ (acting as the base policy π_{θ_0}), a query Q, a temperature T_{sim} for soft assignment, a consistency threshold τ, a regularization coefficient λ_{reg}, and the maximum number of knowledge-update rounds $R_{\max}$. The KL regularization hyperparameters are specified in Algorithm 3.

F.5 Pattern Evolution and Adaptive KL

We present Figure 6 as the evolution record of the IMAP knowledge base; this figure corresponds to the slow knowledge-refresh timescale discussed in Section 3.4, where the adaptive KL controller responds to pattern variance. The x-axis is the knowledge-update round r, while each round contains K_{PPO} PPO steps indexed by k, and each PPO step contains reasoning steps indexed by t. Panel (a) shows task-specific high-frequency Patterns (support count > 100, triggers concentrated on a single task type) and higher-level reusable Patterns (appear across at least two task types and support at least two distinct reasoning schemas). Panel (b) shows merging events, triggered by an LLM-based merge

Algorithm 2 IMAP: Inductive Multi-step Abstraction

Require: Initial LLM $\mathcal{M}$ (policy π_{θ_0}), query Q, soft-assignment temperature T_{sim}, consistency threshold τ, regularization coefficient λ_{reg}, max knowledge-update rounds $R_{\max}$.

Initialize $\mathcal{C}_0 \leftarrow \emptyset$, $\mathcal{P}_0 \leftarrow \emptyset$, ψ_0, $\theta \leftarrow \theta_0$.
for knowledge-update round $r = 0$ to $R_{\max} - 1$ **do**
  Sample reasoning trajectory $\mathrm{CoT}_r \sim \pi_\theta(\cdot \mid Q, s_r)$ for given query Q.
  Set $T_r \leftarrow$ the number of reasoning steps in CoT_r.
  Decompose CoT_r into segments $\{z_i\}$ and induce cases: $\mathcal{C}_{r+1} \leftarrow \mathcal{C}_r \cup \{g_{\text{case}}(z_i)\}$.
  Estimate case embeddings $\{e_c\}_{c \in \mathcal{C}_{r+1}}$.
  if $\mathcal{P}_r \neq \emptyset$ **then**
    Update pattern-assignment parameters ψ by solving: $\psi_{r+1} \leftarrow \arg\max_\psi \mathbb{E}_{c \sim \mathcal{C}_{r+1}} \mathbb{E}_{p \sim q_r(\cdot|c)}[\log P_\psi(p \mid c)] - \lambda_{\text{reg}} \Omega(\psi)$, where $q_r(p \mid c) = P_{\psi_r}(p \mid c)$.
  else
    Set $\psi_{r+1} \leftarrow \psi_r$.
  end if
  if $\mathcal{P}_r \neq \emptyset$ **then**
    Compute consistency score $C_{r,T_r} = \exp(-E(\mathrm{CoT}_{r,\leq T_r}, \mathcal{P}_r))$, where C_{r,T_r} is the final-step consistency score $C_{r,t}$ at $t = T_r$.
  else
    Set $C_{r,T_r} \leftarrow 0$.
  end if
  Generate Pattern set $\mathcal{P}_{r+1}$ by the LLM-based pattern-abstraction operator conditioned on ψ_{r+1} and $\mathcal{C}_{r+1}$.
  Check stopping criterion; **if** satisfied, **break**.
end for
return Pattern set $\mathcal{P}_{R_{\text{fin}}}$ and final knowledge state $s_{R_{\text{fin}}}$.

Table 7: Pattern quality confidence intervals and annotator agreement.

Metric	Mean	95% CI	Agreement
Correctness	92.1%	[89.4, 94.8]	$\kappa = 0.68$
Reusability	4.7	[4.2, 5.0]	$\kappa = 0.61$
Logical completeness	87.3%	[84.1, 90.5]	$\kappa = 0.66$

operator combined with embedding clustering at similarity above 0.85; each merge is followed by a human spot-check, and 46 of 50 spot-checked merges showed no significant loss. Panel (d) reports quality scores from a mixed protocol: correctness and logical completeness are checked automatically on the full Pattern set, while reusability is rated by five annotators on a 1–5 scale for a random sample of 200 Patterns; sampling size and confidence intervals are in Table 7. Panel (e) computes retention rate as the fraction of low-frequency critical constraints (support ≤ 4) that survive after merging; the figure distinguishes retention after post-hoc audit from the frequency bias induced by the training objective.

The adaptive KL controller follows Eq. equation 11 in the main text. High Pattern variance strengthens the KL constraint, whereas low variance relaxes it; the policy remains conservative during knowledge-base transitions. The target KL divergence 0.01 is used as a reference signal. We report a sensitivity analysis over $\lambda_{\max}$ in Table 8.

The ablation variants *w/ Static KL* (fixed λ) and *w/o Adaptive KL* ($\lambda = 0$) isolate different parts of the mechanism.

F.6 Reward Function

$R_{r,t}^{\text{struct}}$ measures structural alignment between the generated reasoning prefix and the pattern set $\mathcal{P}_r$. In our implementation, we set

$$R_{r,t}^{\text{struct}} = \max_{p \in \mathcal{P}_r} \widetilde{\text{sim}}(\mathrm{CoT}_{r,\leq t}, p)$$

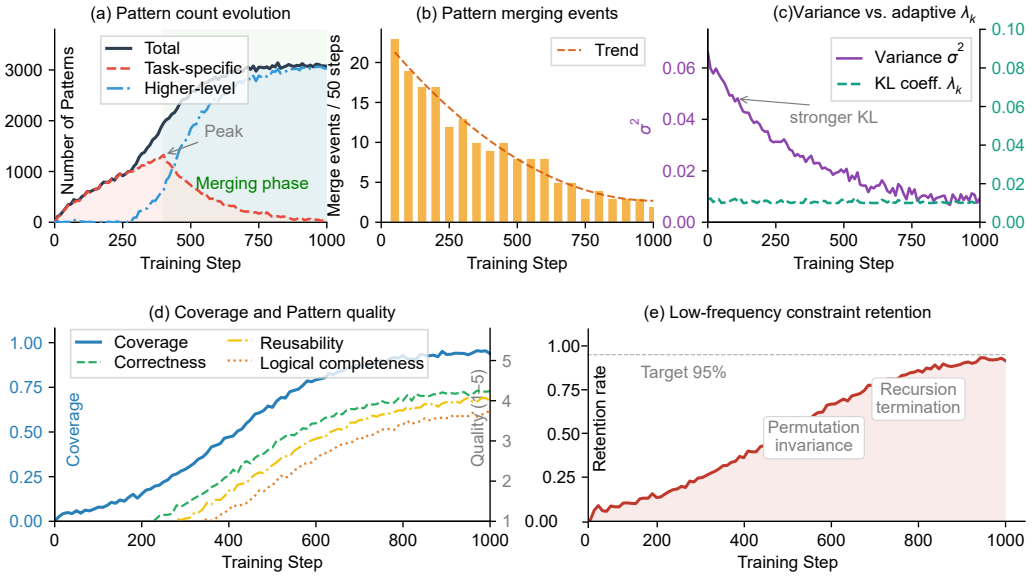


Figure 6: Evolution of the IMAP-KB during training. The x-axis is the knowledge-update round r; each round contains K_{PPO} PPO steps indexed by k, and each PPO step contains reasoning steps indexed by t. (a) Total number of Patterns, task-specific high-frequency Patterns, and higher-level reusable Patterns. Task-specific Patterns are those whose support count exceeds 100 and whose triggers concentrate on a single task type; higher-level reusable Patterns appear across at least two task types and support at least two distinct reasoning schemas. (b) Pattern merging events per 50-step window. Merging is triggered by an LLM-based merge operator combined with embedding clustering at similarity above 0.85. (c) Pattern embedding variance σ^2 and the adaptive KL coefficient $\lambda_k = \min\{\lambda_{\max}, \lambda_{\text{base}} \exp(\eta\sigma^2(\{\bar{\mathbf{p}}_j\}_{j\in W_m(r_k)}))\}$. (d) Pattern coverage and quality scores (correctness, reusability, and logical completeness) on a 1–5 scale. Correctness and logical completeness are checked automatically, while reusability is rated by five annotators. (e) Retention rate of low-frequency but critical constraints, including permutation invariance and recursion termination. Low frequency is defined as support count ≤ 4; retention is computed as the fraction of such constraints that survive after merging. Curves are averaged over five random seeds; shaded regions denote standard deviation.

Table 8: Sensitivity analysis of the adaptive KL controller.

Setting	**Accuracy**	**Train Std.**	**Pattern Var.**
$\lambda_{\max} = 0.1$	35.8%	0.62	0.11
$\lambda_{\max} = 0.5$	36.9%	0.51	0.09
$\lambda_{\max} = 1.0$	37.2%	0.44	0.07

when $\mathcal{P}_r \neq \emptyset$, and $R^{\text{struct}}_{r,t} = 0$ otherwise, where

$$\widetilde{\text{sim}}(x, p) = \frac{1 + \cos(\text{Emb}(x), \text{Emb}(p))}{2} \in [0, 1],$$

and $\text{Emb}(\cdot)$ denotes a fixed sentence embedding function. When $\mathcal{P}_r \neq \emptyset$, $R^{\text{consist}}_{r,t} = C_{r,t} \in (0, 1]$; however, when $\mathcal{P}_r = \emptyset$, $R^{\text{consist}}_{r,t} = 0$ and no consistency penalty is applied. Therefore, the consistency reward $R^{\text{consist}}_{r,t} = C_{r,t}$ provides a graded signal even when verification passes, while the penalty term fires only when $C_{r,t} < \tau$; thus, they serve different roles and are not redundant.

Because Cases and Patterns evolve, $R^{\text{struct}}_{r,t}$ and $R^{\text{consist}}_{r,t}$ are non-stationary across rounds. Consequently, the reward distribution can shift faster than the value function V_ω adapts; as a result, advantage estimates may become biased, and PPO ratios may be miscalibrated. Therefore, we

Algorithm 3 IMAP-RL Training with Closed-Loop PPO Updates

Require: Pretrained policy π_{θ_0}, reference π_{ref}, value function V_ω, initial knowledge state $s_1 = (\mathcal{C}_1, \mathcal{P}_1)$, KL coefficients $\lambda_{\text{base}}, \eta, \lambda_{\max}$, window size m, max knowledge-update rounds $R_{\max}$, PPO update steps per round K_{PPO}, PPO epochs E_{epoch}.

Initialize LoRA parameters Hu et al. (2022) θ, value parameters ω, and auxiliary assignment parameters ψ.
for knowledge-update round $r = 1$ to $R_{\max}$ **do**
  Collect rollout batch with behavior policy π_{old} under $s_r = (\mathcal{C}_r, \mathcal{P}_r)$
  for PPO update step $k = 1$ to K_{PPO} **do**
    Compute Reasonability scores $C_{r,t}$ and rewards $R_{r,t}$ using current $\mathcal{C}_r, \mathcal{P}_r$
    Normalize rewards with running statistics
    Estimate advantages $\hat{A}_{r,t}$ by GAE Schulman et al. (2015) with V_ω
    Compute probability ratio $\rho_{r,t}(\theta)$
    Compute KL coefficient λ_k by Eq. equation 11 using the most recent completed knowledge-update rounds.
    Compute clipped PPO objective $\mathcal{L}_{\text{PPO}}(\theta)$ and KL-regularized total loss $\mathcal{L}_{\text{total}}(\theta)$
    Update θ, ω by clipped PPO for E_{epoch} epochs while π_{old} is frozen
  end for
  Update $\mathcal{C}_{r+1}$ by the LLM-based case-induction operator
  if $\mathcal{P}_r \neq \emptyset$ **then**
    Update ψ_{r+1} by the regularized soft-assignment objective
  else
    Set $\psi_{r+1} \leftarrow \psi_r$
  end if
  Update $\mathcal{P}_{r+1}$ by the LLM-based pattern-abstraction operator
  Set $s_{r+1} = (\mathcal{C}_{r+1}, \mathcal{P}_{r+1})$
  Check stopping criterion; **if** satisfied, **break**
end for
return Final policy π_θ, Cases $\mathcal{C}_{R_{\text{fin}}}$, and Patterns $\mathcal{P}_{R_{\text{fin}}}$

normalize rewards with running statistics and update the knowledge base only every K_{PPO} PPO steps, as detailed in Algorithm 3.

F.7 RL Algorithm Description

Policy Optimization. Algorithm 3 specifies the two-timescale loop: PPO updates occur in the inner loop, and the textual knowledge state is refreshed in the outer loop.

Adaptive KL Regularization. The adaptive KL controller follows Eq. equation 11 in the main text; no separate schedule is introduced here.

F.8 Non-stationary Rewards and PPO Stability

Because Cases and Patterns evolve, $R_{r,t}^{\text{struct}}$ and $R_{r,t}^{\text{consist}}$ are non-stationary. We mitigate this with the two-timescale scheme and the adaptive KL controller described in Appendix F.5; see Section 3.4 for details.

G Experimental Setup

G.1 BBH Evolution

The origins of the BBH dataset can be traced back to this point, and its evolution is shown in detail in Table 9. Specifically, in Srivastava et al. (2022), 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

Table 9: Filtering criteria used to create the BIG-Bench Hard (BBH) subset.

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 10: Decontamination summary. BBH evaluation set contains 6,511 samples across 23 tasks; MATH and GSM8K sizes follow their official releases.

Dataset	Removed	Remaining train	Remaining test
BBH	–	–	6,511 (eval)
MATH	–	7,500	5,000
GSM8K	–	7,473	1,319

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.

G.2 Decontamination

Because BBH, MATH, and GSM8K are public datasets, we perform a decontamination check between training and test splits. The procedure has three stages. First, we remove exact matches after normalization, including whitespace, punctuation, and casing. Second, we compute n-gram overlap ($n = 13$) and remove any training instance whose maximum overlap with a test instance exceeds 0.80; a document is flagged as contaminated if it has at least 3 distinct matching n-grams and a coverage of at least 0.1%. The choice of $n = 13$ follows the decontamination protocol used in prior work OpenAI (2023), which also adopts $n = 13$ for simplicity and consistency across datasets. Third, we compute embedding similarity using a fixed sentence encoder (all-MiniLM-L6-v2 Reimers & Gurevych (2019)) and remove training instances with cosine similarity above 0.90. This threshold follows the semantic-similarity decontamination practice in recent work Yang et al. (2023), where $\tau = 0.9$ is used to capture paraphrased or semantically rewritten samples that might evade n-gram detection. All thresholds are fixed before evaluation. The number of removed instances and the remaining dataset sizes are summarized in Table 10. Moreover, we manually inspect 50 borderline pairs; none are retained if they share the same answer and reasoning structure. Therefore, this decontamination applies to both the main BBH experiments and the cross-domain evaluation. Furthermore, the decontamination results confirm that the observed performance gains stem from generalization rather than benchmark leakage.

G.3 API Utilization Parameters

During the data collection process, we used the Llama-3.1-8B API Grattafiori et al. (2024). We read the terms of service and followed the usage policy. We give the parameter details of the Llama-3.1-8B API used in data collection in Table 11.

G.4 IMAP Training Hyperparameters

The hyperparameters used in our IMAP experiments are described below. The base LLM is `Llama-3.2-1B` for BBH main experiments and `Llama-3.1-8B` for cross-domain experiments Grattafiori et al. (2024). The initial policy π_{θ_0} is the frozen pretrained model $\mathcal{M}$, and the maximum sequence length is 1024. RL fine-tuning follows the adaptive KL-regularized PPO Schulman et al. (2017a) described in Section 3.4. Training runs for 1000 episodes, each episode consisting of up to $T = 5$ reasoning steps, though early termination upon convergence is allowed.

Table 11: Parameters for API utilization.

Parameter	EE	VE	C	Sc	Sy
n	1	3	3	3	1
best-of	1	3	3	3	**2**
model	Llama-3.1-8B	Llama-3.1-8B	Llama-3.1-8B	Llama-3.1-8B	Llama-3.1-8B
temperature	0.9	0.9	0.9	**1**	0.9
max_tokens	128	128	128	128	128
top-p	1	1	1	1	1
frequency-penalty	0	0	0	0	0
presence-penalty	0	0	0	0	0

LoRA. LoRA rank, alpha, and dropout are 16, 32, and 0.05, respectively. The policy learning rate is 1×10^{-5}, the value learning rate is 1×10^{-4}, and the optimizer is AdamW Loshchilov & Hutter (2019).

Pattern Abstraction. The number of Patterns is $K = 20$, the softmax temperature is $\tau_p = 0.5$, the regularization weight is $\lambda_{\text{reg}} = 0.1$, the pattern diversity coefficient is $\nu = 0.05$, and the feature extractor is all-MiniLM-L6-v2 Reimers & Gurevych (2019).

RL Optimization (PPO). The discount factor is $\gamma_{\text{disc}} = 0.99$, the GAE parameter is $\lambda_{\text{GAE}} = 0.95$ Schulman et al. (2015), the PPO clip range is $\epsilon = 0.2$, the number of PPO epochs per update is $E_{\text{epoch}} = 4$, the PPO batch size is $B = 64$, the mini-batch size is $B_{\text{mini}} = 16$, the rollout length is $T_{\max} = 8$, the knowledge refresh interval is $K_{\text{PPO}} = 50$, the total number of episodes is 1000, and the maximum steps per episode is $T = 5$.

Reward Coefficients. The reward coefficients are $w_{\text{acc}} = 1.0$ for accuracy, $w_{\text{struct}} = 0.3$ for structure, $w_{\text{consist}} = 0.2$ for consistency, and $w_{\text{pen}} = 0.1$ for penalty.

Adaptive KL Controller. The base KL factor is $\lambda_{\text{base}} = 0.01$, the maximum KL factor is $\lambda_{\max} = 0.1$, the sensitivity is $\eta = 1.0$, the window size is $m = 5$, and the target KL divergence is 0.01.

Consistency Verification. The threshold is $\tau = 0.5$, the distance metric $d(\cdot, \cdot)$ is cosine distance after Sentence-BERT encoding, and the segmenter g_{case} uses heuristic sentence splitting followed by an LLM parser.

Stopping Criteria. The stopping thresholds are $\epsilon_c = 0.05$ and $\epsilon_p = 0.01$, and evaluation is performed every 100 episodes.

Training. The random seeds are 42, 123, 456, 789, and 1024. Checkpoint selection is based on validation accuracy, and test reporting uses test accuracy at the selected checkpoint.

Compute Resources. We use $1\times$ NVIDIA A100 (40GB) GPU with 16 CPU vCPUs, 40 GB GPU memory, and 64 GB system RAM. Execution time is approximately 19 hours for the 1B model (1000 episodes) and approximately 168 hours for the 8B model (1000 episodes).

The adaptive KL coefficient λ_k is computed as described in Section 3.4. Its dynamics are controlled by the variance of Pattern embeddings over the last m policy iterations. When the Pattern space stabilizes (low variance), λ_k decays to allow policy exploitation; when novel Patterns emerge frequently (high variance), λ_k increases to retain proximity to the reference policy π_{ref}.