

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 HOLON-OF-THOUGHT: IMPROVING ROBUSTNESS IN LARGE LANGUAGE MODELS VIA STRUCTURED FRAMEWORK

006
007 **Anonymous authors**
008 Paper under double-blind review

011 ABSTRACT

013 Large Language Models (LLMs) excel in language comprehension and genera-
014 tion tasks but frequently face challenges in scenarios demanding rigorous logical
015 reasoning or strict adherence to problem conditions. In such reasoning, errors
016 propagate through intermediate steps, hallucinatory outputs violate key problem
017 conditions, and complex problems are often handled in a simplistic, chain-like
018 manner. We propose Holon-of-Thought (**HoT**), a structured reasoning framework.
019 **HoT** explicitly extracts problem conditions and enforces their adherence. It dy-
020 namically decomposes complex problems into verifiable subtasks and solves them
021 through a four-stage pipeline: condition extraction, path exploration, adaptive de-
022 composition, and aggregation. The experimental results show that **HoT** improves
023 the accuracy of the inference and enhances the robustness. This establishes a new
024 paradigm for reliable LLM-based reasoning in mathematics and logic.

027 1 INTRODUCTION

029 The rapid advancement of Large Language Models (LLMs) has driven a transformative shift in
030 artificial intelligence. LLMs now demonstrate strong capabilities in natural language processing,
031 knowledge retrieval, and complex task automation Brown et al. (2020); Naveed et al. (2023). How-
032 ever, despite their impressive breadth of capabilities, LLMs often falter when tasked with rigorous
033 reasoning Wei et al. (2022); Wang et al. (2024a).

034 This brittleness stems from their training paradigm: LLMs learn statistical correlations rather than
035 formal logic, making them prone to errors when faced with problems requiring deductive certainty or
036 strict condition satisfaction Brown et al. (2020); Bender et al. (2021). This shortcoming is especially
037 evident in scenarios requiring strict adherence to logical conditions or precision-oriented decision-
038 making. For example, in mathematical proof generation or engineering design verification, where a
039 single misstep invalidates the entire solution, LLMs often produce outputs that are locally plausible
040 but globally inconsistent First et al. (2023); Lu et al. (2024).

041 In addition, their output often contains hallucinations, which are associated with the neglect of prob-
042 lem conditions or facts Ji et al. (2023); Huang et al. (2025); Zhang et al. (2025). These errors reduce
043 the reliability of LLMs in high-stakes applications, where factual inaccuracies can lead to severe
044 consequences Thirunavukarasu et al. (2023); Dahl et al. (2024); Niu et al. (2024). Hallucinations
045 typically arise when models fill knowledge gaps with statistically plausible but unsubstantiated con-
046 tent Huang et al. (2025); Tommoy et al. (2024).

047 To address these issues, a variety of reasoning-enhancement strategies have been proposed. Among
048 them, Chain-of-Thought (CoT) Wei et al. (2022) prompting has emerged as a widely used approach
049 that encourages models to explicitly enumerate intermediate steps during reasoning. By external-
050 izing the reasoning process, CoT provides a window into the model’s “thinking”, aiding both per-
051 formance and interpretability Kojima et al. (2022). Although CoT improves performance on many
052 multistep problems, it still exhibits brittleness: it can overlook hard conditions, generate invalid in-
053 termediate steps Wang et al. (2023); Arcuschin et al. (2025). This underscores the need for more
structured, condition-aware reasoning frameworks to achieve robust and interpretable reasoning.

054 For the sake of robust and interpretable reasoning, we propose Holon-of-Thought (**HoT**). **HoT** is a
 055 structured reasoning framework with a four-stage pipeline of condition extraction, path exploration,
 056 adaptive decomposition, and aggregation, explicitly enforcing condition adherence and dynamically
 057 decomposing problems into verifiable subtasks. The focus on exploration and condition satisfaction
 058 reflects the strategies used in combinatorial optimization and automated planning.

059 This approach draws on classical condition satisfaction systems and modern adaptive computation
 060 techniques, balancing thoroughness with efficiency Graves (2017). **HoT**’s architecture is designed to
 061 be model-agnostic, operating purely through prompt engineering or lightweight API calls, ensuring
 062 wide applicability without retraining overhead.

063 **HoT** innovates by explicitly extracting and enforcing both explicit and implicit problem conditions,
 064 generating and scoring multiple high-level solution paths to select the optimal one, adaptively de-
 065 composing complex problems into isolated verifiable subtasks based on complexity, and aggregating
 066 sub-solutions while ensuring global condition adherence. These innovations offer three key advan-
 067 tages: (1) robust reasoning through explicit condition prioritization, reducing error propagation; (2)
 068 improved interpretability through structured, auditable reasoning traces; and (3) computational ef-
 069 ficiency through selective reasoning, enabled by pruning—generating multiple candidate methods
 070 and retaining only the optimal one, thus avoiding wasteful exploration of dead ends.

071 Our work underscores that achieving robust reasoning in LLMs requires condition-aware architec-
 072 tural designs that prioritize structure, precision, and verifiability. **HoT** exemplifies this principle
 073 by promoting a disciplined approach to reasoning. The structured methodology enables LLMs to
 074 reason more conservatively and avoid compounding errors, especially in tasks where correctness is
 075 tightly coupled with condition satisfaction. By combining selective exploration with rigorous syn-
 076 thesis, **HoT** provides a scalable blueprint for deploying LLMs in engineering applications where
 077 reliability and interpretability are paramount.

079 2 RELATED WORK

080 Prompt-based reasoning aims to unlock the complex capabilities of LLMs without expensive fine-
 081 tuning. The paradigm was pioneered by CoT prompting, which generates intermediate steps to trace
 082 a sequential reasoning process Wei et al. (2022). This concept was extended by methods like Tree
 083 of Thoughts Yao et al. (2023a) and Graph of Thoughts Besta et al. (2024), which explore non-linear
 084 reasoning paths using more expressive tree and graph structures, respectively, to manage complex
 085 problem solving. In contrast, **HoT** differentiates itself by integrating upfront condition extraction
 086 and adaptive decomposition into its path exploration.

087 The field has since expanded rapidly, with research exploring numerous avenues to enhance LLM’s
 088 reasoning. Many efforts have focused on iterative refinement, where models critique and improve
 089 their own outputs, such as Self-Refine Madaan et al. (2023), Step-Back Zheng et al. (2024) and
 090 System 2 Attention Weston & Sukhbaatar (2023). Other approaches incorporate external formalisms
 091 to add rigor. For example, Logical Thoughts Zhao et al. (2023) integrates symbolic logic, while
 092 other methods use structured formats like symbolic expressions, tables, or executable code to offload
 093 computation and enforce syntactic correctness Hu et al. (2024); Wang et al. (2024b); Puerto et al.
 094 (2024); Gao et al. (2023). These methods demonstrate the diverse strategies being investigated to
 095 make LLM’s reasoning more powerful and reliable. In contrast to these iterative or formalism-based
 096 techniques, **HoT** focuses on grounding reasoning in extracted conditions and dynamically adapting
 097 the problem structure. **HoT** offers a framework that complements these methods by emphasizing
 098 condition enforcement without repeated critiques or extensive external tools.

099 A significant challenge is ensuring the robustness of generated reasoning, as standard CoT is of-
 100 ten susceptible to process errors or hallucinations. To address this, Self-Consistency Wang et al.
 101 (2023) and LLM-Blender Jiang et al. (2023) mitigate errors via multitrajectory consensus, though
 102 this brute-force approach incurs high computational costs. EchoPrompt Mekala et al. (2024) seeks
 103 efficiency by distilling divergent rationales into a unified path but risks reinforcing errors if initial
 104 paths are flawed. This reveals a core tension: aggregating diverse paths for robustness can be either
 105 computationally expensive or risk converging on an incorrect solution. Some technologies avoid
 106 this problem by using prompt words. For example, “specify constraints pattern” Moundas et al.
 107 (2024) was proposed to process constraints and reduce noise interference, but this method only op-

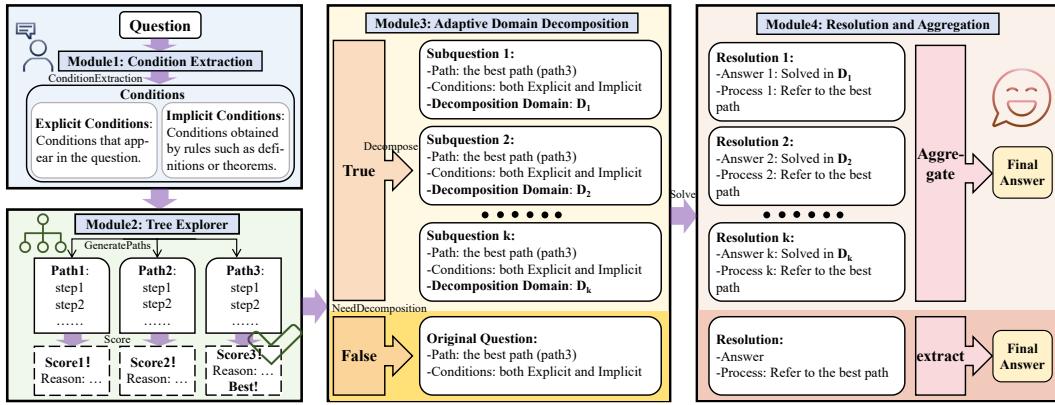


Figure 1: **HoT** Framework. It includes four modules: Condition Extraction, Tree Explorer, Adaptive Domain Decomposition, and Resolution and Aggregation. Condition Extraction identifies both explicit and implicit conditions from the original problem. Tree Explorer generates three potential solution paths and selects the one with the highest score as the basis for subsequent reasoning. Adaptive Domain Decomposition determines whether the problem should be decomposed, based on its complexity and the extracted conditions. If decomposition is necessary (as illustrated in the top half of the diagram), the problem is split into sub-problems. Resolution and Aggregation then solves each sub-problem individually and combines their results to generate the final answer. If decomposition is not needed (as shown in the bottom half), Resolution and Aggregation directly solves the original problem and outputs the final result.

erates at the level of data annotation and ignores the crucial role of implicit conditions in reasoning. Contrastive Chain-of-Thought Prompting Chia et al. (2023) provides both positive and negative exemplar reasoning chains to guide the model away from common mistakes, improving reasoning quality in a structured way. Contrastive Denoising with Noisy Chain-of-Thought Zhou et al. (2024) constructs noisy-rationale scenarios and learns to denoise rationales by contrasting noisy and clean ones. Chain-of-Defensive-Thought Wang et al. (2025) uses structured, defensive reasoning exemplars to enhance robustness. These methods improve robustness but often at the cost of flexibility, computational efficiency, or general applicability.

Another critical research direction tackles the rigidity of linear reasoning through two intertwined strategies: problem decomposition and process adaptability. For decomposition, methods like Thread-of-Thought (ThoT) Zhou et al. (2023) segment complex inputs, while architectural innovations like Layer-of-Thoughts (LoT) Fungwacharakorn et al. (2024) impose predefined hierarchies. However, their static nature limits effectiveness: ThoT relies on brittle fixed segmentation heuristics, and LoT's rigid schemas may misalign with task logic. Other methods focus more on process adaptability. For instance, Buffer of Thoughts (BoT) Yang et al. (2024) uses memory to reuse prior reasoning segments but prioritizes past solutions over adapting structure for novel tasks. Recursive Decomposition of Logical Thought (RDoLT) Qasim et al. (2025) recursively decomposes tasks with scoring and knowledge propagation but faces scalability limits due to computational overhead. Instance-Adaptive Prompting (IAP) Yuan et al. (2024) adapts the prompt based on input complexity, but this only tunes the starting point, leaving the subsequent single-path reasoning flow unchanged. This highlights the need for a framework that can dynamically structure its reasoning process, flexibly decomposing the problem as needed, and does not require too much computational resources.

3 METHODOLOGY

We design the **HoT** framework to significantly enhance the reasoning robustness of LLMs by explicitly extracting problem conditions and systematically exploring the solution space. **HoT** guides the model toward logically grounded reasoning paths and mitigates error propagation across complex problem decompositions. As shown in Figure 1, the **HoT** framework contains four modules: Condition Extraction, Tree Explorer, Adaptive Domain Decomposition, Resolution and Aggregation.

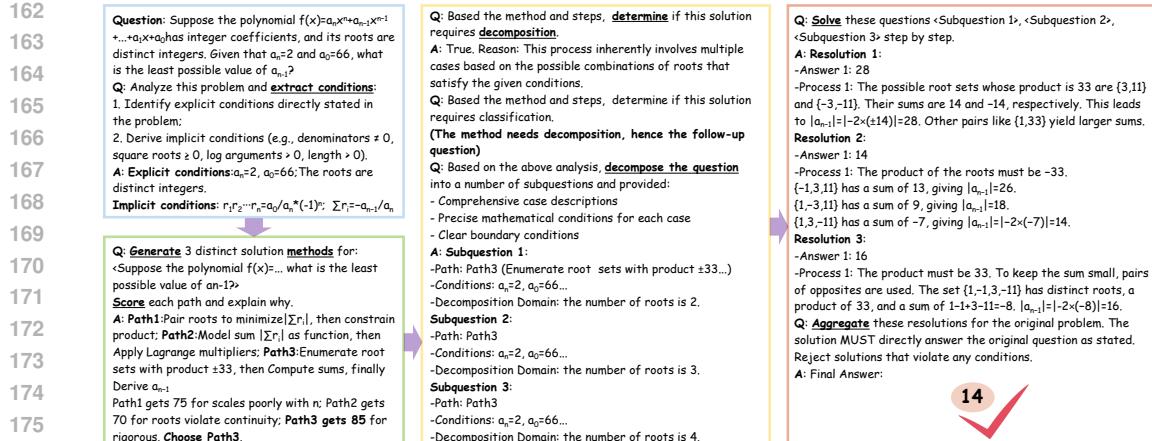


Figure 2: Example of **HoT** Prompting in Solving a Complex Math Problem. This figure corresponds one-to-one with Figure 1, illustrating the practical execution of each module and flow in the **HoT** framework. It demonstrates the application of the **HoT** framework to a polynomial root problem. It shows the decomposition of the original problem into subproblems based on the number of roots, the resolution of each subproblem using the selected optimal path (Path 3), and the aggregation of results to derive the final answer. This example validates **HoT**’s effectiveness in handling multi-case mathematical reasoning through structured decomposition and verification.

3.1 CONDITION EXTRACTION

HoT utilizes Condition Extractor module to transform raw question statements into a structured representation by identifying explicit and implicit conditions. This module acts as the “grounding phase,” forcing the LLM to explicitly articulate the rules and boundaries governing the problem before attempting solutions, addressing a common failure mode where models overlook implicit conditions. It structures the conditions and transforms the fuzzy input into a computable framework, helping prevent failure in subsequent steps due to missing information. Its output acts as a shared, immutable condition set referenced throughout the **HoT** pipeline.

In our definition, conditions C are divided into explicit conditions C_e and implicit conditions C_i . C_e are directly parsed from mathematical formulations or logical statements, while C_i are derivable through mathematical or contextual rules or common sense knowledge relevant to the domain (e.g., “ages must be positive integers”, “a triangle’s angles sum to 180 degrees”).

Given a question Q , we get C_e and C_i :

$$\text{ConditionExtraction}_{\text{LLM}}(Q) \rightarrow (C_e, C_i) \quad (1)$$

For example, given the question Q : “What is the sum of the three digit cubes that are the cubes of either squares or cubes?” We can distill the explicit conditions, “The cubes must be of numbers that are either squares or cubes themselves”, and the implicit conditions, “The three-digit cubes range from 100 to 999, so the cube roots must be integers between 5 and 9 inclusive, because $4^3=64$ (too small) and $10^3=1000$ (too large)”. The distillation of these conditions will directly guide subsequent work.

3.2 TREE EXPLORER

HoT uses Tree Explorer module to determine a path to solve the question. Tree Explorer module firstly generates a set of viable solution paths, $\mathcal{P} = \{\pi_1, \pi_2, \dots, \pi_N\}$:

$$\text{GeneratePaths}_{\text{LLM}}(Q, C_e, C_i) \rightarrow \mathcal{P} \quad (2)$$

Each path π_i represents a high-level strategy, detailing the proposed method and concrete implementation steps. This is not a search through intermediate steps, but a generation of complete, end-to-end strategies. By generating multiple paths, the model explores a diverse range of question-solving paradigms before committing to a single approach.

Once multiple paths are proposed, the next phase involves a rigorous evaluation to select the most promising one.

3.2.1 PATH SCORING

Each path π_i is systematically evaluated by LLM based on a predefined set of criteria, such as anticipated accuracy, operational feasibility, and computational complexity. The model assigns a Heuristic score, $s_i \in [0, 100]$ to each path, resulting in a set of scored tuples:

$$\mathcal{P}_{\text{scored}} = \{(\pi_i, s_i) \mid s_i = \text{Score}_{\text{LLM}}(\pi_i, Q, \mathcal{C}_e, \mathcal{C}_i)\} \quad (3)$$

3.2.2 OPTIMAL PATH SELECTION

The path with the highest score is selected as the optimal strategy, π^* .

$$\pi^* = \underset{\pi_i \in \mathcal{P}}{\operatorname{argmax}} s_i \quad (4)$$

Optimal Path Selection is a decisive step that dictates the entire subsequent execution flow.

Tree Explorer introduces a critical self-reflection step, enabling the model to deliberate on the quality of its own plans before execution. Also, Tree Explorer provides a classification basis for Adaptive Domain Decomposition.

3.3 ADAPTIVE DOMAIN DECOMPOSITION

Adaptive Domain Decomposition module determines the final execution strategy by assessing the question's complexity relative to the chosen path π^* . It decides whether a divide-and-conquer approach is necessary.

First, the framework performs a binary classification to determine if the question domain should be partitioned:

$$\begin{aligned} \text{DecomposeFlag} &= \text{NeedDecomposition}_{\text{LLM}}(\pi^*) \\ &\in \{\text{True}, \text{False}\} \end{aligned} \quad (5)$$

When DecomposeFlag is True, **HoT** partitions the problem into logically isolated subproblems $\mathcal{Q}_{\text{sub}} = \{Q_1, Q_2, \dots, Q_k\}$. This decomposition strategically splits the feasible domain defined by \mathcal{C}_e and \mathcal{C}_i , aligning subproblem boundaries with critical decision points in π^* . Each subproblem inherits relevant conditions and operates in semantically isolated containers—ensuring errors in one subdomain cannot propagate to others. This approach transforms complex combinatorial, multi-case, or recursive problems into parallelizable verification tasks while maintaining strict condition adherence.

$$\mathcal{Q}_{\text{sub}} = \text{Decompose}_{\text{LLM}}(Q, \pi^*, \mathcal{C}_e, \mathcal{C}_i) \quad (6)$$

When DecomposeFlag is False, the problem does not need to be decomposed and can be solved directly into the next Module (see Direct Resolution for Resolution and Aggregation).

3.4 RESOLUTION AND AGGREGATION

Resolution and Aggregation module executes the plan established in Adaptive Domain Decomposition module. The reasoning process follows one of two pathways based on the outcome of the adaptive decomposition. This bifurcation ensures computational efficiency for simple problems while maintaining rigorous error isolation for complex ones, adapting dynamically to the problem's needs.

- Direct Resolution: when DecomposeFlag is False, the question is considered monolithic. The model applies the chosen strategy π^* to solve the original question Q in a single, direct pass while ensuring the solution satisfies \mathcal{C}_e and \mathcal{C}_i . This path is typical for problems with short reasoning chains or those where decomposition would introduce unnecessary overhead (e.g., single-step arithmetic, straightforward logical inferences). The final solution is obtained as:

$$A = \text{Solve}_{\text{LLM}}(Q, \pi^*, \mathcal{C}_e, \mathcal{C}_i) \quad (7)$$

270 • Hierarchical Resolution: when *DecomposeFlag* is True, the model engages in a multi-step
 271 hierarchical process. First, it independently resolves each subquestion $Q_i \in \mathcal{Q}_{sub}$ using
 272 the logic of π^* , yielding a set of sub-answers $\mathcal{A}_{sub} = \{A_1, A_2, \dots, A_k\}$, while ensuring all
 273 solutions strictly adhere to \mathcal{C}_e and \mathcal{C}_i . Each subquestion is solved in isolation, preventing
 274 error propagation between subproblems. For instance, in a problem requiring case analysis
 275 (e.g., “Solve for x where $x^2 + bx + c = 0$, considering discriminant cases”), each subproblem
 276 corresponds to a distinct case ($D>0$, $D=0$, $D<0$), and solving one case incorrectly does
 277 not affect others. For the answer to each subquestion, there is the following relationship
 278 equation:

279
$$A_i = \text{Solve}_{LLM}(Q_i, \pi^*, \mathcal{C}_e, \mathcal{C}_i), \forall Q_i \in \mathcal{Q}_{sub} \quad (8)$$

280 Next, these partial solutions are synthesized. The model aggregates the information from
 281 \mathcal{A}_{sub} to construct a single, coherent, and comprehensive final solution and confirm that the
 282 aggregated answer satisfies all conditions, \mathcal{C}_e and \mathcal{C}_i , directly answers the original question
 283 Q. Aggregation rules are problem-specific. For summation problems, it might involve sim-
 284 ple addition; for case analysis, logical combination; for condition satisfaction, intersection
 285 of valid solutions.

286
$$A = \text{Aggregate}_{LLM}(\mathcal{A}_{sub}, Q, \mathcal{C}_e, \mathcal{C}_i) \quad (9)$$

287 Figure1 illustrates this dynamic pipeline, highlighting how conditional execution optimizes the
 288 trade-off between thoroughness (for complex tasks) and efficiency (for simpler ones). **Algorithm1**
 289 shows the **HoT** reasoning process in a formal description. Figure2 illustrates an example of solving
 290 a complex math problem using **HoT** framework. Here, since *DecomposeFlag* is True, the problem
 291 undergoes decomposition and aggregation, and the correct answer is obtained.

292 This conditional execution allows **HoT** to dynamically adapt its strategy, applying a more robust,
 293 multi-step reasoning process only when necessary, thereby optimizing for both accuracy and effi-
 294 ciency. Finally, the resulting answer A is formatted for the end-user.

297 4 EXPERIMENTS

299 To rigorously evaluate the proposed **HoT**, we conducted a comprehensive set of experiments de-
 300 signed to assess its performance, generalizability, robustness, and the contribution of its core com-
 301 ponents. Our evaluation demonstrates that **HoT** achieves superior accuracy on a diverse suite of
 302 mathematical (GSM8K, ASDiv, SVAMP) and logical (OpenBookQA, Strategy) reasoning bench-
 303 marks. We further show that these performance gains are model-agnostic, enhancing the capabilities
 304 of multiple underlying LLMs. Critically, through quantitative stability metrics, we found that **HoT**
 305 not only provides more accurate results but does so with significantly greater consistency and lower
 306 variance than baseline methods. Finally, a detailed ablation study confirmed that each module of
 307 **HoT** is integral to its success, with its structured approach of identifying conditions, decomposing
 308 problems, and exploring solution paths being fundamental to its effectiveness.

309 4.1 EXPERIMENTAL SETUP

311 4.1.1 DATASETS.

313 We evaluate **HoT** with two types of datasets, Math: GSM8K Cobbe et al. (2021), ASDiv Miao et al.
 314 (2021), SVAMP Patel et al. (2021), and Logic: OpenBookQA Mihaylov et al. (2018), StrategyQA
 315 Geva et al. (2021). These datasets share common characteristics: a certain depth of thought and the
 316 need to synthesize knowledge and reasoning. For each dataset we take the first 200 examples of the
 317 test set.

318 4.1.2 MODELS.

320 We use Qwen2.5:7b-instruct Qwen et al. (2025) as the backbone model for our main experiments
 321 due to its superior semantic comprehension and execution capabilities. We also use DeepSeek-V3
 322 DeepSeek-AI et al. (2025) to test the effectiveness of **HoT** and complete robustness testing via API.
 323 All the experiments on Qwen2.5:7b-instruct are run on an 1x NVIDIA A100 GPU server. The
 324 temperature hyperparameter T of models is set to 0.3.

324 4.1.3 BASELINES.
325

326 Our baselines include: CoT Wei et al. (2022), Random-CoT Fu et al. (2022), CoT with Self-
327 Consistency (CoT-SC) Wang et al. (2023), ReAct Yao et al. (2023b), instance-adaptive prompting
328 strategy (IAP) Yuan et al. (2024) and AoT Teng et al. (2025). For CoT-SC, we set the number of
329 paths $n = 5$. For ReAct, we designed similarly styled prompts for each dataset as examples. For IAP,
330 we adopt the Majority Vote strategy as our approach. Accuracy is the average value of the results of
331 3 runs, and detailed reproduction settings are provided in Appendix A.6.

332 4.1.4 METRICS.
333

334 We adopt both standard and newly designed metrics to evaluate different reasoning methods. In
335 the main experiments and model comparison studies, we report the average accuracy over three in-
336 dependent runs. For robustness testing, we design two complementary metrics based on repeated
337 testing: (1) **Total Variance (TV)**, and (2) **Instance Variance Mean (IVM)**. These metrics are de-
338 fined in a general form to allow application across various experimental settings. Then we introduce
339 how to obtain the two metrics.

340 Let each method be tested over M independent runs. Each run consists of N problems. The total
341 runs are grouped into B problem blocks, and each problem block contains N problems. Each
342 problem block is evaluated R times. Thus, the total number of runs is $M = R \times B$.

343 Let $A_b^{(r)}$ denote the accuracy of the model on the b -th problem block in its r -th repetition, where
344 $b \in \{1, 2, \dots, B\}$, and $r \in \{1, 2, \dots, R\}$. The mapping from the pair (b, r) to the global run index
345 i is given by $i = (b - 1) \times R + r$.

346 Let $A^{(i)}$ denote the accuracy of the i -th run (where $i \in \{1, 2, \dots, M\}$) and the mapping from (b, r)
347 to i is mentioned above). Then:

$$349 \text{TV} = \text{Var}(A^{(1)}, A^{(2)}, \dots, A^{(M)}) \\ 350$$

351 For each block b , we compute the variance of the accuracy values across the R repetitions:
352

$$353 V_b = \text{Var}(A_b^{(1)}, A_b^{(2)}, \dots, A_b^{(R)}). \\ 354$$

355 IVM is then defined as the average variance across all B problem blocks:

$$356 \text{IVM} = \frac{1}{B} \sum_{b=1}^B V_b \\ 357 \\ 358$$

360 4.2 EXPERIMENTAL RESULTS AND ANALYSIS
361

362 4.2.1 MAIN RESULTS.

363 As shown in Table1, **HoT** achieves excellent performance across all datasets. On SVAMP, it achieves
364 91.6% accuracy (+0.4% over AoT); on OpenBookQA, it reaches 91.5% accuracy (+3.9% over CoT-
365 SC). **HoT** * is derived from voting among CoT, HoT, and HoT without Tree Explorer (Module 2),
366 with **HoT**'s answer as the tiebreaker. **HoT** * further improves performance: 92.2% on GSM8K
367 (+2.0% over AoT) and 92.2% on average (+1.2% over **HoT**). Result show that **HoT** is able to
368 improve the accuracy of LLM reasoning whether solving mathematical or logical problems. With
369 more computational resources, **HoT** can continue to improve LLMs' inference ability.

370 4.2.2 REASONING MODELS COMPARISON RESULTS.
371

372 We evaluate the effectiveness of the proposed **HoT** framework across two representative reason-
373 ing benchmarks (GSM8K and OpenBookQA) , using two LLMs with different parameter scales:
374 Qwen2.5:7b-Instruct and DeepSeek-V3. Table2 shows that, **HoT** consistently improves perfor-
375 mance across datasets. For Qwen2.5:7b-instruct, **HoT** boosts accuracy from 81.6% to 86.3% on
376 GSM8K (+4.7%) and from 83.9% to 91.5% on OpenBookQA (+7.6%). For DeepSeek-V3, per-
377 formance improves from 91.3% to 97.0% on GSM8K (+5.7%) and from 94.5% to 96.7% on Open-
378 BookQA (+2.2%). The larger relative gains for Qwen2.5:7b-instruct suggest that **HoT** is particularly

Methods	GSM8K	ASDiv	SVAMP	OpenBookQA	StrategyQA	Avg.
CoT	81.8	93.2	86.1	83.9	84.8	86.0
Random-CoT	82.6	94.1	88.3	82.5	87.3	87.0
CoT-SC (n=5)	85.1	94.4	90.8	87.6	88.0	89.2
ReAct	84.1	94.7	90.9	87.1	85.1	88.4
IAP	80.5	91.1	87.2	75.3	<u>88.8</u>	84.6
AoT	<u>90.2</u>	<u>97.2</u>	91.2	84.9	86.5	90.0
HoT (Ours)	86.3	<u>97.2</u>	<u>91.6</u>	91.5	88.1	<u>91.0</u>
HoT * (Ours)	92.2	97.7	93.8	<u>88.3</u>	89.3	92.2

Table 1: Performance Comparison of Reasoning Methods Across Datasets (%). This table compares the accuracy of **HoT** and **HoT *** with baseline methods (CoT, Random-CoT, CoT-SC, ReAct, IAP) on five math and logic reasoning datasets. The experiment was conducted on the Qwen2.5:7b-instruct model. Results show **HoT** achieves a high average accuracy (91.0%) and **HoT *** further sets a new state-of-the-art (92.2%), confirming their superiority in mathematical and logical reasoning tasks.

Methods	GSM8K	OpenBookQA	Avg.
Qwen2.5:7b-instruct	CoT	81.6	83.9
	HoT	86.3	91.5
DeepSeek-V3	CoT	<u>91.3</u>	<u>94.5</u>
	HoT	97.0	96.7

Table 2: **HoT**’s Performance Across Different LLM Backbones (%). This table evaluates **HoT**’s effectiveness on two LLMs (Qwen2.5:7b-instruct and DeepSeek-V3). Evaluations are conducted on GSM8K (mathematics) and OpenBookQA (logic). Results indicate **HoT** consistently improves reasoning accuracy for both LLMs, demonstrating its model-agnostic ability to enhance reasoning in both small and large parameter LLMs.

valuable for improving the reasoning capabilities of smaller LLMs, while the consistent performance boosts for DeepSeek-V3 demonstrate that strong models can also benefit from **HoT**’s structured resolution. These results highlight **HoT**’s general effectiveness across LLM’s scales and task types.

4.2.3 ROBUSTNESS TESTING.

Robustness testing is performed using Total Variance (TV) and Instance Variance Mean (IVM) metrics on 300 runs with parameters M=300, N=10, B=100 and R=3, comparing the stability of **HoT** with CoT. We selected GSM8K as the dataset. CoT serves as the baseline with a Total Variance (TV) of 144.5 and an Instance Variance Mean (IVM) of 34.7. CoT-SC shows a higher TV of 155.0 (+7.27%) but a lower IVM of 22.7 (-34.58%). In contrast, **HoT** achieves a lower TV of 98.0 (-32.18%) and an IVM of 19.1 (-44.96%). Visual analysis in Figure3 further confirms that **HoT** exhibits substantially improved stability in inference results. TV quantifies global variability across multiple problem blocks. **HoT**’s lower TV indicates stronger robustness regardless of input distribution. IVM measures robustness by averaging variance in accuracy across repeated trials on the same problem blocks. The reduction in IVM highlights **HoT**’s ability to produce consistent results for identical problems, alleviating the unstable output that often troubles unstructured and unreliable reasoning methods. For comparisons of the remaining datasets, see the Appendix A.2.

4.2.4 ABLATION STUDY.

Ablation Studies have proved the effectiveness of each module. As shown in Table3, when **HoT** removes Module1 and does not explicitly mention the conditions in any of the subsequent treatments, there is the most significant drop in performance; when **HoT** removes Module2 and does not explore methods and paths, there is a slight drop in performance; when **HoT** removes Mod-

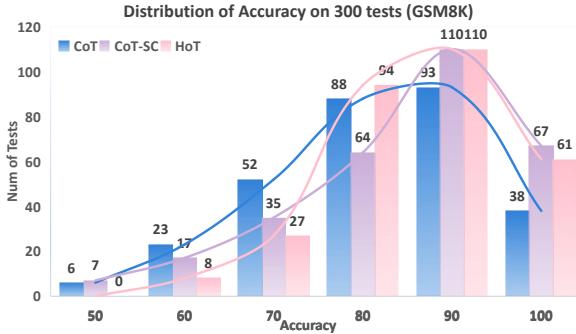


Figure 3: Robustness Comparison Between CoT, CoT-SC and **HoT** on the GSM8K Dataset. This figure presents the distribution of accuracy across 300 test runs for CoT, CoT-SC and **HoT** methods. The experiment was conducted on the Qwen2.5:7b-instruct model. It visually demonstrates that **HoT** exhibits significantly lower variance in inference results compared to CoT and CoT-SC, confirming **HoT**’s superior stability and consistency in repeated reasoning tasks.

Methods	GSM8K	OpenBookQA
HoT (Full)	86.3	91.5
HoT (w/o Module1)	82.2	89.5
HoT (w/o Module2)	<u>85.5</u>	89.7
HoT (w/o Module3, 4)	82.7	<u>91.2</u>

Table 3: Ablation Study on **HoT** Core Components (%). This table assesses the contribution of each **HoT** module by removing them individually. Results show removing Module 1 (Condition Extraction) causes the largest performance drop (4.1% on GSM8K, 2.0% on OpenBookQA), while removing other modules leads to smaller declines. This confirms that all modules are integral to **HoT**’s success, with condition extraction being critical for maintaining reasoning accuracy.

ule3, 4 and does not conduct problem decomposition and aggregation, there is also a slight drop in performance. Without condition extraction and maintenance, the model sometimes ignores important information when reasoning, leading to biased results. Without the exploration of methods and paths, the model has a more inadequate understanding of the problem and lacks an understanding of the big picture. Decomposing and aggregating the problem can reduce the complexity of the demand solution problem. Since subproblems often require only linear reasoning (without the need to consider multiple scenarios in parallel), this allows for different treatments in different situations without contaminating each other, further improving the robustness of the solution.

5 CONCLUSION

HoT represents a pivotal advance in enabling reliable, condition-aware reasoning in LLMs. It integrates explicit condition extraction, strategic planning, adaptive decomposition, and structured aggregation into a cohesive prompt-based framework, directly addressing core limitations of existing methods—including error propagation, poor interpretability, and weak condition satisfaction. Experiments validate **HoT**’s effectiveness: it boosts performance across diverse LLMs, and demonstrates greater robustness. Its structured reasoning traces enhance auditability, making it ideal for high-stakes fields like engineering and decision support.

HoT’s success underscores a key insight: advancing LLM reasoning requires reimagining the process, not just scaling models or data. Future work will extend **HoT** to dynamic/uncertain conditions, integrate external subproblem verifiers, and apply it to real-world engineering tasks. Automating decomposition heuristics (tailored to problem and condition structure) is another promising direction. By providing a blueprint for structured, condition-aware reasoning, **HoT** lays the groundwork for more trustworthy LLMs in rigorous domains.

486 REFERENCES
487

488 Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and
489 Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful, 2025. URL <https://arxiv.org/abs/2503.08679>.

490

491 Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the
492 dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM*
493 *conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

494

495 Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawska, Lukas Gian-
496 inazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczek, et al. Graph of
497 thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI*
498 *conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

499

500 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
501 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
502 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

503

504 Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. Contrastive chain-
505 of-thought prompting, 2023. URL <https://arxiv.org/abs/2311.09277>.

506

507 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
508 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
509 Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.

510

511 Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. Large legal fictions: Profiling legal
512 hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93, 2024.

513

514 DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Cheng-
515 gang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang,
516 Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting
517 Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui
518 Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi
519 Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li,
520 Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang,
521 Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun
522 Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan
523 Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J.
524 Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang,
525 Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shaoqing Wu, Shengfeng
526 Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiiping Yu, Shunfeng Zhou, Shut-
527 ing Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao,
528 Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue
529 Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xi-
530 aokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin
531 Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang,
532 Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang
533 Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui
534 Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying
535 Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu,
536 Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan
537 Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F.
538 Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda
539 Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao,
540 Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li,
541 Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL
542 <https://arxiv.org/abs/2412.19437>.

543

544 Emily First, Markus N Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and
545 repair with large language models. In *Proceedings of the 31st ACM Joint European Software*

540 *Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1229–
 541 1241, 2023.

542 Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting
 543 for multi-step reasoning. *arXiv preprint arXiv:2210.00720*, 2022.

544 Wachara Fungwacharakorn, Nguyen Ha Thanh, May Myo Zin, and Ken Satoh. Layer-of-thoughts
 545 prompting (lot): Leveraging llm-based retrieval with constraint hierarchies, 2024. URL <https://arxiv.org/abs/2410.12153>.

546 Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and
 547 Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine
 548 Learning*, pp. 10764–10799. PMLR, 2023.

549 Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle
 550 use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of
 551 the Association for Computational Linguistics*, 9:346–361, 2021.

552 Alex Graves. Adaptive computation time for recurrent neural networks, 2017. URL <https://arxiv.org/abs/1603.08983>.

553 Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. Chain-of-
 554 symbol prompting for spatial reasoning in large language models. In *First Conference on Lan-
 555 guage Modeling*, 2024.

556 Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong
 557 Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language
 558 models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information
 559 Systems*, 43(2):1–55, 2025.

560 Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang,
 561 Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM
 562 computing surveys*, 55(12):1–38, 2023.

563 Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models
 564 with pairwise ranking and generative fusion, 2023. URL <https://arxiv.org/abs/2306.02561>.

565 Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large
 566 language models are zero-shot reasoners. *Advances in neural information processing systems*,
 567 35:22199–22213, 2022.

568 Minghai Lu, Benjamin Delaware, and Tianyi Zhang. Proof automation with large language models.
 569 In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engi-
 570 neering*, pp. 1509–1520, 2024.

571 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri
 572 Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement
 573 with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

574 Rajasekhar Reddy Mekala, Yasaman Razeghi, and Sameer Singh. Echoprompt: Instructing the
 575 model to rephrase queries for improved in-context learning, 2024. URL <https://arxiv.org/abs/2309.10687>.

576 Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing
 577 english math word problem solvers, 2021. URL <https://arxiv.org/abs/2106.15772>.

578 Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct
 579 electricity? a new dataset for open book question answering, 2018. URL <https://arxiv.org/abs/1809.02789>.

580 Max Moundas, Jules White, and Douglas C Schmidt. Prompt patterns for structured data extraction
 581 from unstructured text. In *Proceedings of the 31st Pattern Languages of Programming (PLoP
 582 Conference (Columbia River Gorge, WA)*, 2024.

594 Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman,
 595 Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language
 596 models. *ACM Transactions on Intelligent Systems and Technology*, 2023.

597 Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. Mitigating hallucinations in
 598 large language models via self-refinement-enhanced knowledge retrieval, 2024. URL <https://arxiv.org/abs/2405.06545>.

600 Arkil Patel, Satwik Bhattacharya, and Navin Goyal. Are nlp models really able to solve simple math
 601 word problems?, 2021. URL <https://arxiv.org/abs/2103.07191>.

602 Haritz Puerto, Martin Tutek, Somak Aditya, Xiaodan Zhu, and Iryna Gurevych. Code prompting
 603 elicits conditional reasoning abilities in text+code llms, 2024. URL <https://arxiv.org/abs/2401.10065>.

604 Kaleem Ullah Qasim, Jiashu Zhang, Tariq Alsahfi, and Ateeq Ur Rehman Butt. Recursive decompo-
 605 sition of logical thoughts: Framework for superior reasoning and knowledge propagation in large
 606 language models, 2025. URL <https://arxiv.org/abs/2501.02026>.

607 Qwen, ;, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 608 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 609 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 610 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 611 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 612 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.
 613 URL <https://arxiv.org/abs/2412.15115>.

614 Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. Atom of thoughts
 615 for markov llm test-time scaling, 2025. URL <https://arxiv.org/abs/2502.12018>.

616 Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez,
 617 Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*,
 618 29(8):1930–1940, 2023.

619 SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das.
 620 A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv*
 621 preprint [arXiv:2401.01313](https://arxiv.org/abs/2401.01313), 6, 2024.

622 Wenxiao Wang, Parsa Hosseini, and Soheil Feizi. Chain-of-defensive-thought: Structured reasoning
 623 elicits robustness in large language models against reference corruption, 2025. URL <https://arxiv.org/abs/2504.20769>.

624 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 625 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models,
 626 2023. URL <https://arxiv.org/abs/2203.11171>.

627 Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Georgiev, Rocktim Jyoti
 628 Das, and Preslav Nakov. Factuality of large language models: A survey, 2024a. URL <https://arxiv.org/abs/2402.02420>.

629 Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang,
 630 Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. Chain-of-table: Evolving
 631 tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*, 2024b.

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

635 Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too),
 636 2023. URL <https://arxiv.org/abs/2311.11829>.

637 Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez,
 638 and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models.
 639 *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024.

648 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik
 649 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Ad-*
 650 *vances in neural information processing systems*, 36:11809–11822, 2023a.

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

654 Xiaosong Yuan, Chen Shen, Shaotian Yan, Xiaofeng Zhang, Liang Xie, Wenxiao Wang, Renu
 655 Guan, Ying Wang, and Jieping Ye. Instance-adaptive zero-shot chain-of-thought prompting. *Ad-*
 656 *vances in Neural Information Processing Systems*, 37:125469–125486, 2024.

657 Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao,
 658 Yu Zhang, Yulong Chen, et al. Siren’s song in the ai ocean: A survey on hallucination in large
 659 language models. *Computational Linguistics*, pp. 1–45, 2025.

660 Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter.
 661 Enhancing zero-shot chain-of-thought reasoning in large language models through logic. *arXiv*
 662 *preprint arXiv:2309.13339*, 2023.

663 Huaxiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le,
 664 and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models,
 665 2024. URL <https://arxiv.org/abs/2310.06117>.

666 Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and
 667 Jianbing Shen. Thread of thought unraveling chaotic contexts, 2023. URL <https://arxiv.org/abs/2311.08734>.

668 Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language
 669 models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances*
 670 *in Neural Information Processing Systems*, 37:123846–123910, 2024.

671

672 A APPENDIX

673 A.1 EXAMPLE OF HOT REASONING PROCESS

674 A.1.1 QUESTION

675 Suppose the polynomial

$$676 f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

677 has integer coefficients, and its roots are distinct integers.

678 Given that $a_n = 2$ and $a_0 = 66$, what is the least possible value of $|a_{n-1}|$?

679 A.1.2 SOLUTION

680 Since $f(x)$ has integer coefficients, the Integer Root Theorem tells us that all integer roots of $f(x)$
 681 must divide the constant term $66 = 2 \cdot 3 \cdot 11$. Thus, the possible integer roots of $f(x)$ are

$$682 \pm 1, \pm 2, \pm 3, \pm 6, \pm 11, \pm 22, \pm 33, \pm 66.$$

683 Moreover, since we know that all roots of $f(x)$ are integers, we know that all roots of $f(x)$ appear
 684 in the list above.

685 Now we apply Vieta’s formulas. The product of the roots of $f(x)$ is $(-1)^n \cdot \frac{a_0}{a_n}$, which is 33 or -33.
 686 Also, the sum of the roots is $-\frac{a_{n-1}}{a_n} = -\frac{a_{n-1}}{2}$. Thus, in order to minimize $|a_{n-1}|$, we should make
 687 the absolute value of the sum of the roots as small as possible, working under the constraint that the
 688 product of the roots must be 33 or -33.

689 We now consider two cases.

702 Case 1 is that one of 33, -33 is a root, in which case the only other possible roots are ± 1 . In this
 703 case, the absolute value of the sum of the roots is at least 32.

704 The alternative, Case 2, is that one of 11, -11 is a root and one of 3, -3 is a root. Again, the only
 705 other possible roots are ± 1 , so the absolute value of the sum of the roots is at least $11 - 3 - 1 = 7$,
 706 which is better than the result of Case 1. If the absolute value of the sum of the roots is 7, then
 707 $|a_{n-1}| = 7|a_n| = 7 \cdot 2 = 14$.

708 Therefore, we have shown that $|a_{n-1}| \geq 14$, and we can check that equality is achieved by

$$710 \quad f(x) = 2(x + 11)(x - 3)(x - 1) \\ 711 \quad = 2x^3 + 14x^2 - 82x + 66,$$

712 which has integer coefficients and integer roots. So the least possible value of $|a_{n-1}|$ is 14.

713 A.1.3 IO (INPUT/OUTPUT) METHOD

714 According to Vieta's formulas, the product of the roots r_1, r_2, \dots, r_n is related to the coefficients
 715 by the formula $(-1)^n r_1 r_2 \cdots r_n = \frac{a_0}{a_n} = \frac{66}{2} = 33$. The coefficient a_{n-1} is given by $a_{n-1} =$
 716 $-a_n \sum_{i=1}^n r_i = -2 \sum_{i=1}^n r_i$. To minimize $|a_{n-1}|$, we must minimize the absolute value of the sum
 717 of the roots. We need to find a set of distinct integer roots whose product is ± 33 . The integer factors
 718 of 33 are $\pm 1, \pm 3, \pm 11, \pm 33$.

719 To make the sum of the roots as close to zero as possible, we should select roots that cancel each
 720 other out. Consider the set of four distinct integer roots $\{1, -1, 3, -11\}$. The product of these roots
 721 is $(1)(-1)(3)(-11) = 33$. For this set of roots, the sum is $1 + (-1) + 3 + (-11) = -8$. This
 722 corresponds to a value of $a_{n-1} = -2 \times (-8) = 16$. Therefore, the least possible value of $|a_{n-1}|$ is
16.

723 A.1.4 HOT METHOD

724 CONDITION EXTRACTION

725 The first module of the process focuses on meticulously extracting and defining all constraints from
 726 the problem statement. The explicit constraints are that the polynomial $f(x)$ has **integer coefficients**,
 727 its roots are **distinct integers**, the leading coefficient is $a_n = 2$, and the constant term is
 728 $a_0 = 66$.

729 From these, several implicit constraints are derived. The relationship between coefficients and roots
 730 is governed by Vieta's formulas. This leads to two critical deductions:

- 731 The product of the roots, adjusted for the leading coefficient, is fixed. Specifically,
 $(-1)^n \prod_{i=1}^n r_i = \frac{a_0}{a_n} = \frac{66}{2} = 33$. This means the product of the n distinct integer roots
 732 must be either 33 (if n is even) or -33 (if n is odd).
- 733 The coefficient a_{n-1} is directly proportional to the sum of the roots: $a_{n-1} =$
 $-a_n \sum_{i=1}^n r_i = -2 \sum_{i=1}^n r_i$.

734 The ultimate goal is to find the minimum possible value of $|a_{n-1}|$, which translates to finding a set
 735 of distinct integer roots that satisfies the product constraint while making the absolute value of their
 736 sum as small as possible.

737 TREE EXPLORER

738 The second module explores various potential strategies for solving the problem. Three primary
 739 methods were identified and scored based on their perceived effectiveness and rigor:

- 740 **Algebraic Approach (Score: 85):** This method, deemed the most promising, involves a
 $\text{741 systematic application of Vieta's formulas. The core idea is to enumerate all possible sets}$
 $\text{742 of distinct integer roots whose product is } \pm 33, \text{ calculate the sum for each set, determine}$
 $\text{743 the corresponding } |a_{n-1}|, \text{ and identify the minimum value. Its strength lies in its direct and}$
 $\text{744 exhaustive nature.}$

756 • **Combinatorial Optimization (Score: 75):** This approach frames the problem as mini-
 757 758 759 760 761 762 763 764
 765 766 767 768 769 770 771 772 773
 774 775 776 777 778 779 780 781 782 783 784
 785 786 787 788 789 790 791 792 793 794
 795 796 797 798 799 800 801 802 803 804 805
 806 807 808 809
 810

- **Combinatorial Optimization (Score: 75):** This approach frames the problem as minimizing the absolute sum of a set of integers under the constraint that their product is fixed. While conceptually sound, it was considered slightly less direct than the algebraic approach because it focuses more on the optimization aspect rather than a structured enumeration based on the factors of 33.
- **Analytical Approach (Score: 70):** This method proposed using calculus to treat a_{n-1} as a function of the roots and find its minimum. This was rated the lowest because the problem's domain consists of discrete integers, making continuous calculus methods difficult and impractical to apply directly without significant adaptation.

The algebraic approach was selected as the most direct and reliable path to the solution.

ADAPTIVE DOMAIN DECOMPOSITION

The third module takes the chosen algebraic approach and breaks the problem down into a series of smaller, manageable subproblems or cases. The primary decomposition is based on the number of distinct integer roots, n . Since the product of the roots must be ± 33 , and $33 = 3 \times 11$, the number of roots is not predetermined but must be at least two. The analysis was therefore structured to check cases for different values of n .

The subproblems identified were:

775 • **Case $n = 2$ (Two roots):** The product of the roots, $r_1 r_2$, must be 33. All pairs of distinct
 776 777 778 779 780 781 782 783 784
 785 786 787 788 789 790 791 792 793 794
 795 796 797 798 799 800 801 802 803 804
 805 806 807 808 809
 810

- **Case $n = 2$ (Two roots):** The product of the roots, $r_1 r_2$, must be 33. All pairs of distinct integers with this product are examined.
- **Case $n = 3$ (Three roots):** The product of the roots, $r_1 r_2 r_3$, must be -33 . All unique triplets of distinct integers with this product are investigated.
- **Case $n = 4$ (Four roots):** The product, $r_1 r_2 r_3 r_4$, must be 33. All quartets of distinct integers satisfying this are considered.
- **Cases $n > 4$:** These cases are also considered, but it's reasoned that adding more roots, especially pairs like $(1, -1)$, is the most efficient way to increase n while controlling the sum.

SUBPROBLEM RESOLUTION AND AGGREGATION

The final module involves solving each subproblem defined in the previous stage and then aggregating the results to find the overall minimum value.

795 • **Solving for $n=2$:** The possible root sets whose product is 33 are $\{3, 11\}$ and $\{-3, -11\}$.
 796 797 798 799 800 801 802 803 804 805
 806 807 808 809
 810

- **Solving for $n=2$:** The possible root sets whose product is 33 are $\{3, 11\}$ and $\{-3, -11\}$. Their sums are 14 and -14 , respectively. This leads to $|a_{n-1}| = |-2 \times (\pm 14)| = 28$. Other pairs like $\{1, 33\}$ yield larger sums.
- **Solving for $n=3$:** The product of the roots must be -33 . The system systematically explores combinations:
 - $\{-1, 3, 11\}$ has a sum of 13, giving $|a_{n-1}| = 26$.
 - $\{1, -3, 11\}$ has a sum of 9, giving $|a_{n-1}| = 18$.
 - $\{1, 3, -11\}$ has a sum of -7 , giving $|a_{n-1}| = |-2 \times (-7)| = 14$.
 - $\{-1, -3, 11\}$ has a product of 33, not -33 , so it's invalid for $n = 3$.
- **Solving for $n=4$:** The product must be 33. To keep the sum small, pairs of opposites are used. The set $\{1, -1, 3, -11\}$ has distinct roots, a product of 33, and a sum of $1 - 1 + 3 - 11 = -8$. This yields $|a_{n-1}| = |-2 \times (-8)| = 16$.
- **Aggregation and Final Answer:** The results from all valid cases are compared:
 - For $n=2$, the minimum $|a_{n-1}|$ is 28.
 - For $n=3$, the minimum $|a_{n-1}|$ is 14.
 - For $n=4$, the minimum $|a_{n-1}|$ is 16.

The overall minimum value found across all examined cases is 14. This occurs for the set of three distinct integer roots $\{1, 3, -11\}$. The reasoning correctly accounts for the parity of n affecting the sign of the root product, a detail the other methods missed, leading them to incorrect conclusions.

Final Answer: 14

810 **Algorithm 1** **HoT** Reasoning Algorithm

811 **Input:** Initial question Q

812 **Output:** Final answer A

813 1: $\mathcal{C}_e, \mathcal{C}_i \leftarrow \text{ConditionExtraction}_{\text{LLM}}(Q)$
 814 //Extract explicit/implicit conditions

815 2: $\mathcal{P} \leftarrow \text{GeneratePaths}_{\text{LLM}}(Q, \mathcal{C}_e, \mathcal{C}_i)$
 816 //Generate solution paths

817 3: $\pi^* \leftarrow \underset{\pi_i \in \mathcal{P}}{\text{argmax}} \text{Score}_{\text{LLM}}(\pi_i)$
 818 //Select optimal path

819 4: $\text{DecomposeFlag} \leftarrow \text{NeedDecomposition}_{\text{LLM}}(\pi^*)$

820 5: **if** $\text{DecomposeFlag} == \text{True}$ **then**

821 6: $\mathcal{Q}_{\text{sub}} \leftarrow \text{Decompose}_{\text{LLM}}(Q, \pi^*, \mathcal{C}_e, \mathcal{C}_i)$
 822 //Domain decomposition

823 7: $\mathcal{A}_{\text{sub}} \leftarrow \emptyset$

824 8: **for** $Q_i \in \mathcal{Q}_{\text{sub}}$ **do**

825 9: $A_i \leftarrow \text{Solve}_{\text{LLM}}(Q_i, \pi^*, \mathcal{C}_e, \mathcal{C}_i)$

826 10: $\mathcal{A}_{\text{sub}} \leftarrow \mathcal{A}_{\text{sub}} \cup \{A_i\}$

827 11: **end for**

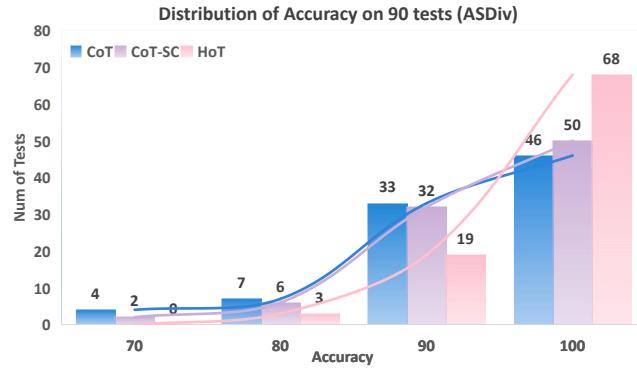
828 12: $A \leftarrow \text{Aggregate}_{\text{LLM}}(\mathcal{A}_{\text{sub}}, Q, \mathcal{C}_e, \mathcal{C}_i)$

829 13: **else**

830 14: $A \leftarrow \text{Solve}_{\text{LLM}}(Q, \pi^*, \mathcal{C}_e, \mathcal{C}_i)$
 831 //Direct resolution

832 15: **end if**

833 16: **return** A



847 Figure 4: Robustness Comparison Between CoT, CoT-SC and **HoT** on the ASDiv Dataset. This
 848 figure presents the distribution of accuracy across 90 test runs for CoT, CoT-SC and **HoT** methods.
 849 The experiment was conducted on the Qwen2.5:7b-instruct model.

850

851 **A.2 COMPLETE ROBUSTNESS TESTING**

852

853 **A.2.1 ROBUSTNESS TESTING ON ASDIV DATASET**

854 Robustness testing on ASDiv Dataset is performed on 90 runs with parameters $M=90$, $N=10$, $B=30$
 855 and $R=3$. CoT shows a TV of 64.8 and an IVM of 9.3. CoT-SC shows a TV of 51.4 (-20.68%)
 856 and an IVM of 7.9 (-15.05%). **HoT** achieves lower variance with a TV of 26.7 (-58.80%) and an
 857 IVM of 8.6 (-7.73%). Figure4 illustrates the trend.

858

859 **A.2.2 ROBUSTNESS TESTING ON SVAMP DATASET**

860

861 Robustness testing on SVAMP Dataset is performed on 90 runs with parameters $M=90$, $N=10$, $B=30$
 862 and $R=3$. CoT shows a TV of 117.1 and an IVM of 17.0. CoT-SC shows a TV of 82.7 (-29.38%)
 863 and an IVM of 14.8 (-12.94%). **HoT** achieves lower variance with a TV of 79.8 (-31.85%) and an
 IVM of 15.6 (-8.24%). Figure5 illustrates the trend.

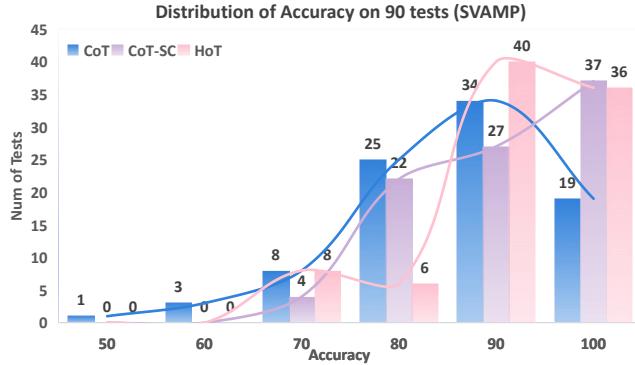


Figure 5: Robustness Comparison Between CoT, CoT-SC and **HoT** on the SVAMP Dataset. This figure presents the distribution of accuracy across 90 test runs for CoT, CoT-SC and **HoT** methods. The experiment was conducted on the Qwen2.5:7b-instruct model.

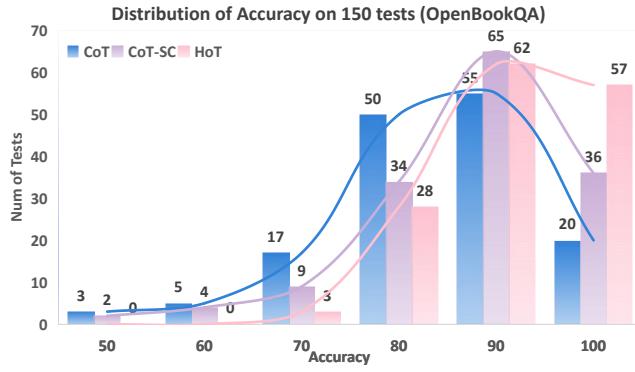


Figure 6: Robustness Comparison Between CoT, CoT-SC and **HoT** on the OpenBookQA Dataset. This figure presents the distribution of accuracy across 150 test runs for CoT, CoT-SC and **HoT** methods. The experiment was conducted on the Qwen2.5:7b-instruct model.

A.2.3 ROBUSTNESS TESTING ON OPENBOOKQA DATASET

Robustness testing on OpenBookQA Dataset is performed on 150 runs with parameters $M=150$, $N=10$, $B=50$ and $R=3$. CoT shows a TV of 117.2 and an IVM of 16.0. CoT-SC shows a TV of 110.2 (-5.97%) and an IVM of 14.7 (-8.13%). **HoT** achieves lower variance with a TV of 62.3 (-46.84%) and an IVM of 10.2 (-36.25%). Figure6 illustrates the trend.

A.2.4 ROBUSTNESS TESTING ON STRATEGYQA DATASET

Robustness testing on StrategyQA Dataset is performed on 300 runs with parameters $M=300$, $N=10$, $B=100$ and $R=3$. CoT shows a TV of 124.3 and an IVM of 32.2. CoT-SC shows a TV of 103.8 (-16.49%) and an IVM of 20.7 (-35.71%). **HoT** achieves lower variance with a TV of 87.2 (-29.85%) and an IVM of 18.2 (-43.48%). Figure7 illustrates the trend.

A.3 COST ANALYSIS

This section quantifies the trade-off between performance and efficiency of **HoT** and its variant, compared to baseline reasoning methods. The cost analysis utilized the results from Table1 and the corresponding average resource consumption values calculated for each individual question. To address large disparities in token consumption across methods, the token count is visualized on a log10 scale. This ensures clear separation of data points

Figure8 presents the relationship between accuracy and computational resource consumption for **HoT**, **HoT** *, and six baseline methods (CoT, Random-CoT, CoT-SC, ReAct, IAP, AoT). **HoT**

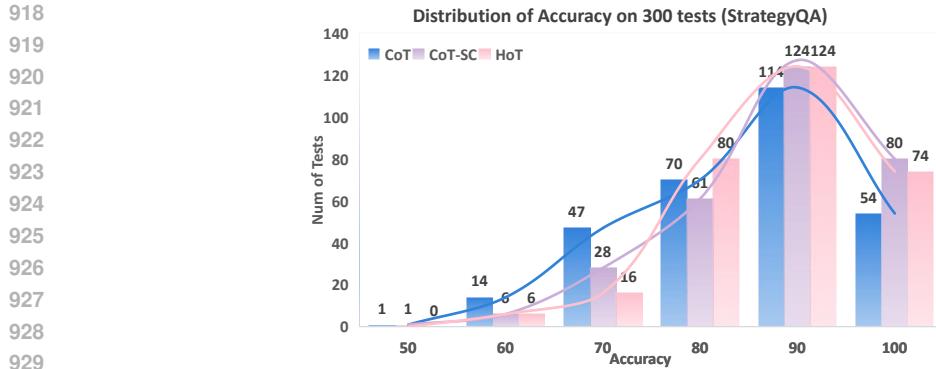


Figure 7: Robustness Comparison Between CoT, CoT-SC and **HoT** on the StrategyQA Dataset. This figure presents the distribution of accuracy across 300 test runs for CoT, CoT-SC and **HoT** methods. The experiment was conducted on the Qwen2.5:7b-instruct model.

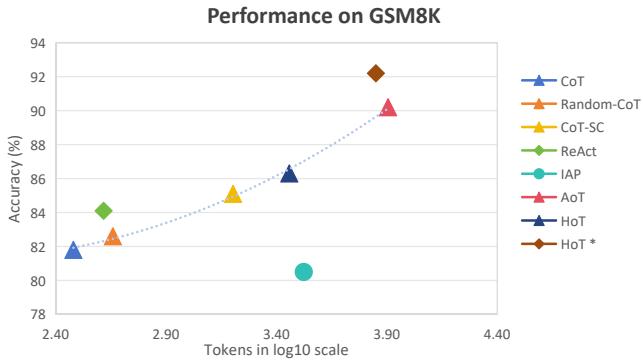


Figure 8: Accuracy vs. Computational Resource Consumption (Total Token Count in log10 Scale) Across Reasoning Methods

strikes a balance between accuracy and efficiency, aligning with the growth trend observed in most baseline methods. Meanwhile, **HoT** * demonstrates excellent performance by significantly boosting accuracy while increasing computational resource consumption.

A.4 PROMPTS OF **HoT** IN MATHEMATICS

In this section, we introduce prompts for each module of **HoT** in Mathematics (GSM8K).

A.4.1 CONDITION EXTRACTION

```

959 1  async def _condition_extraction(self, question: str) -> Dict[str, Any]:
960 2      prompt = f"""
961 3          You are a world-class mathematician and mathematical logician.
962 4          You are intelligent, rigorous, and cautious.
963 5          You always reason step by step, consider all relevant conditions.
964 6          You think in terms of structure, symmetry, and mathematical
965 7              principles, and never skip important logical steps.
966 8          You aim to find a complete and correct solution, not just an
967 9              answer.
968 10         You THINK CLEARLY, STRUCTURALLY, AND DEEPLY.
969 11         Analyze this math problem and extract ALL conditions:
970 12         problem:{question}
971 13         Notice:
972 14         1. Identify explicit conditions (directly stated in the problem)
973 15         2. Derive implicit conditions (e.g., denominators > 0, square
974 16             roots > 0, log arguments > 0)

```

```
972 14     3. Determine domain restrictions based on mathematical principles
973 15     4. Identify range limitations from problem context
974 16     5. Extract physical meaning conditions (e.g., length > 0,
975 17         probability in [0,1])
976 18     Output JSON format:
977 19     {{
978 20         "explicit": ["condition1", "condition2"],
979 21         "implicit": ["condition1", "condition2"],
980 22         "notes": "Additional analysis notes"
981 23     }}
982 24     """
983 25     for attempt in range(self.config.max_retries):
984 26         try:
985 27             response = await self.llm.generate(prompt, response_format="
986 28                 json_object")
987 29             data = json.loads(response)
988 30             if not isinstance(data, dict):
989 31                 continue
990 32             conditions = {
991 33                 "explicit": data.get("explicit", []),
992 34                 "implicit": data.get("implicit", []),
993 35                 "notes": data.get("notes", "")}
994 36             if not (conditions["explicit"] or conditions["implicit"]):
995 37                 continue
996 38             return conditions
997 39         except (json.JSONDecodeError, AttributeError) as e:
998 40             continue
999 41     return {
1000 42         "explicit": ["Default explicit condition"],
1001 43         "implicit": ["Default implicit condition"],
1002 44         "notes": "Fallback conditions"
1003 45     }
```

Listing 1: Condition Extraction Prompt in Mathematics

A.4.2 TREE EXPLORER

```
1  async def _tree_explorer(self, question: str) -> List[Dict[str, Any]]:
2      prompt = f"""
3          You are a world-class mathematician and mathematical logician.
4          You are intelligent, rigorous, and cautious.
5          You always reason step by step, consider all relevant conditions.
6          You think in terms of structure, symmetry, and mathematical
7              principles, and never skip important logical steps.
8          You aim to find a complete and correct solution, not just an
9              answer.
10         You THINK CLEARLY, STRUCTURALLY, AND DEEPLY.
11         Generate 3 distinct solution methods for:
12         {question}
13         Notice:
14         1. Employ different theoretical frameworks (algebraic, geometric,
15             analytical, etc.)
16         2. Approach from fundamentally different perspectives
17         3. Vary implementation techniques significantly
18         4. Consider both conventional and innovative methods
19         5. Steps can be retained as ideas only, without exact
20             calculations
21         6. Pay attention to the mathematical expressions in the questions
22             and understand them correctly
23         7. examine carefully the subject matter
24         For each method, provide:
25             - Clear description of the mathematical approach
26             - Step-by-step implementation plan
```

```

1026      - Effectiveness score (0-100) based on:
1027      * Mathematical rigor
1028      * Computational feasibility
1029      * Logical completeness
1030      * Efficiency
1031
1032      Output JSON format:
1033      {{ "methods": [
1034          {{ "description": "Method description",
1035              "steps": ["step1", "step2"],
1036              "score": 0-100,
1037              "score_reason": "Scoring justification"
1038          } }
1039      ] }
1040      """
1041
1042      for attempt in range(self.config.max_retries):
1043          try:
1044              response = await self.llm.generate(prompt, response_format="json_object")
1045              response = response.strip()
1046              data = json.loads(response)
1047              if not isinstance(data, dict) or "methods" not in data:
1048                  raise ValueError("Invalid structure: missing 'methods' key")
1049              methods = data["methods"]
1050              if len(methods) != 3:
1051                  raise ValueError(f"Expected 3 methods, got {len(methods)}")
1052              required_keys = {"description", "steps", "score", "score_reason"}
1053              for method in methods:
1054                  if not all(k in method for k in required_keys):
1055                      raise ValueError("Missing required keys in method")
1056                  if not isinstance(method["steps"], list):
1057                      raise ValueError("Steps must be a list")
1058              return sorted(methods, key=lambda x: -x["score"])
1059          except (json.JSONDecodeError, ValueError, KeyError) as e:
1060              if attempt == self.config.max_retries - 1:
1061                  return []
1062              continue
1063      return []
1064
1065
1066
1067      A.4.3 ADAPTIVE DOMAIN DECOMPOSITION
1068
1069      1 async def _adaptive_domain_decomposition(self, method: str, steps: List[str]) -> Dict[str, Any]:
1070          2 prompt = f"""
1071          3             You are a world-class mathematician and mathematical logician.
1072          4             You are intelligent, rigorous, and cautious.
1073          5             You always reason step by step, consider all relevant conditions.
1074          6             You think in terms of structure, symmetry, and mathematical
1075          7                 principles, and never skip important logical steps.
1076          8             You aim to find a complete and correct solution, not just an
1077          9                 answer.
1078          10            You THINK CLEARLY, STRUCTURALLY, AND DEEPLY.
1079          11            Determine if this solution requires classification:
1080          12            Method: {method}
1081          13            Steps: {steps}

```

Listing 2: Tree Explorer Prompt in Mathematics

```

1067      A.4.3 ADAPTIVE DOMAIN DECOMPOSITION
1068
1069      1 async def _adaptive_domain_decomposition(self, method: str, steps: List[str]) -> Dict[str, Any]:
1070          2 prompt = f"""
1071          3             You are a world-class mathematician and mathematical logician.
1072          4             You are intelligent, rigorous, and cautious.
1073          5             You always reason step by step, consider all relevant conditions.
1074          6             You think in terms of structure, symmetry, and mathematical
1075          7                 principles, and never skip important logical steps.
1076          8             You aim to find a complete and correct solution, not just an
1077          9                 answer.
1078          10            You THINK CLEARLY, STRUCTURALLY, AND DEEPLY.
1079          11            Determine if this solution requires classification:
1080          12            Method: {method}
1081          13            Steps: {steps}

```

```

1080
1081     12
1082     13     Notice:
1083     14     1. Identify parameter dependencies requiring discussion
1084     15     2. Detect interval-specific elements (absolute values, piecewise
1085     16     functions)
1086     17     3. Recognize domain-specific computation requirements
1087     18     4. Flag multiple solution sets needing verification
1088     19     5. Pay attention to the mathematical expressions in the questions
1089     20     and understand them correctly
1090     21     6. examine carefully the subject matter
1091     22     If classification needed, provide:
1092     23     - Comprehensive case descriptions
1093     24     - Precise mathematical conditions for each case
1094     25     - Clear boundary conditions
1095     26     Output JSON format:
1096     27     {{{
1097     28         "need_classify": true/false,
1098     29         "reason": "Classification rationale",
1099     30         "cases": [
1100     31             {{{
1101     32                 "description": "Case description",
1102     33                 "conditions": [{"parameter": "value_range"}]
1103     34             }}}
1104     35         ]}}
1105     36     }}}
1106     37     """
1107     38     response = await self.llm.generate(prompt, response_format="json_object")
1108     39     try:
1109     40         data = json.loads(response)
1110     41         return {
1111     42             "need_classify": data.get("need_classify", False),
1112     43             "reason": data.get("reason", ""),
1113     44             "cases": data.get("cases", [])
1114     45         }
1115     46     except json.JSONDecodeError:
1116     47         return {"need_classify": False, "reason": "Parse failed", "cases": []}
1117     48
1118     49     Listing 3: Adaptive Domain Decomposition Prompt in Mathematics
1119     50
1120     51
1121     52
1122     53
1123     54
1124     55
1125     56
1126     57
1127     58
1128     59
1129     60
1130     61
1131     62
1132     63
1133     64
1134     65
1135     66
1136     67
1137     68
1138     69
1139     70
1140     71
1141     72
1142     73
1143     74
1144     75
1145     76
1146     77
1147     78
1148     79
1149     80
1150     81
1151     82
1152     83
1153     84
1154     85
1155     86
1156     87
1157     88
1158     89
1159     90
1160     91
1161     92
1162     93
1163     94
1164     95
1165     96
1166     97
1167     98
1168     99
1169     100
1170     101
1171     102
1172     103
1173     104
1174     105
1175     106
1176     107
1177     108
1178     109
1179     110
1180     111
1181     112
1182     113
1183     114
1184     115
1185     116
1186     117
1187     118
1188     119
1189     120
1190     121
1191     122
1192     123
1193     124
1194     125
1195     126
1196     127
1197     128
1198     129
1199     130
1200     131
1201     132
1202     133
1203     134
1204     135
1205     136
1206     137
1207     138
1208     139
1209     140
1210     141
1211     142
1212     143
1213     144
1214     145
1215     146
1216     147
1217     148
1218     149
1219     150
1220     151
1221     152
1222     153
1223     154
1224     155
1225     156
1226     157
1227     158
1228     159
1229     160
1230     161
1231     162
1232     163
1233     164
1234     165
1235     166
1236     167
1237     168
1238     169
1239     170
1240     171
1241     172
1242     173
1243     174
1244     175
1245     176
1246     177
1247     178
1248     179
1249     180
1250     181
1251     182
1252     183
1253     184
1254     185
1255     186
1256     187
1257     188
1258     189
1259     190
1260     191
1261     192
1262     193
1263     194
1264     195
1265     196
1266     197
1267     198
1268     199
1269     200
1270     201
1271     202
1272     203
1273     204
1274     205
1275     206
1276     207
1277     208
1278     209
1279     210
1280     211
1281     212
1282     213
1283     214
1284     215
1285     216
1286     217
1287     218
1288     219
1289     220
1290     221
1291     222
1292     223
1293     224
1294     225
1295     226
1296     227
1297     228
1298     229
1299     230
1300     231
1301     232
1302     233
1303     234
1304     235
1305     236
1306     237
1307     238
1308     239
1309     240
1310     241
1311     242
1312     243
1313     244
1314     245
1315     246
1316     247
1317     248
1318     249
1319     250
1320     251
1321     252
1322     253
1323     254
1324     255
1325     256
1326     257
1327     258
1328     259
1329     260
1330     261
1331     262
1332     263
1333     264
1334     265
1335     266
1336     267
1337     268
1338     269
1339     270
1340     271
1341     272
1342     273
1343     274
1344     275
1345     276
1346     277
1347     278
1348     279
1349     280
1350     281
1351     282
1352     283
1353     284
1354     285
1355     286
1356     287
1357     288
1358     289
1359     290
1360     291
1361     292
1362     293
1363     294
1364     295
1365     296
1366     297
1367     298
1368     299
1369     300
1370     301
1371     302
1372     303
1373     304
1374     305
1375     306
1376     307
1377     308
1378     309
1379     310
1380     311
1381     312
1382     313
1383     314
1384     315
1385     316
1386     317
1387     318
1388     319
1389     320
1390     321
1391     322
1392     323
1393     324
1394     325
1395     326
1396     327
1397     328
1398     329
1399     330
1400     331
1401     332
1402     333
1403     334
1404     335
1405     336
1406     337
1407     338
1408     339
1409     340
1410     341
1411     342
1412     343
1413     344
1414     345
1415     346
1416     347
1417     348
1418     349
1419     350
1420     351
1421     352
1422     353
1423     354
1424     355
1425     356
1426     357
1427     358
1428     359
1429     360
1430     361
1431     362
1432     363
1433     364
1434     365
1435     366
1436     367
1437     368
1438     369
1439     370
1440     371
1441     372
1442     373
1443     374
1444     375
1445     376
1446     377
1447     378
1448     379
1449     380
1450     381
1451     382
1452     383
1453     384
1454     385
1455     386
1456     387
1457     388
1458     389
1459     390
1460     391
1461     392
1462     393
1463     394
1464     395
1465     396
1466     397
1467     398
1468     399
1469     400
1470     401
1471     402
1472     403
1473     404
1474     405
1475     406
1476     407
1477     408
1478     409
1479     410
1480     411
1481     412
1482     413
1483     414
1484     415
1485     416
1486     417
1487     418
1488     419
1489     420
1490     421
1491     422
1492     423
1493     424
1494     425
1495     426
1496     427
1497     428
1498     429
1499     430
1500     431
1501     432
1502     433
1503     434
1504     435
1505     436
1506     437
1507     438
1508     439
1509     440
1510     441
1511     442
1512     443
1513     444
1514     445
1515     446
1516     447
1517     448
1518     449
1519     450
1520     451
1521     452
1522     453
1523     454
1524     455
1525     456
1526     457
1527     458
1528     459
1529     460
1530     461
1531     462
1532     463
1533     464
1534     465
1535     466
1536     467
1537     468
1538     469
1539     470
1540     471
1541     472
1542     473
1543     474
1544     475
1545     476
1546     477
1547     478
1548     479
1549     480
1550     481
1551     482
1552     483
1553     484
1554     485
1555     486
1556     487
1557     488
1558     489
1559     490
1560     491
1561     492
1562     493
1563     494
1564     495
1565     496
1566     497
1567     498
1568     499
1569     500
1570     501
1571     502
1572     503
1573     504
1574     505
1575     506
1576     507
1577     508
1578     509
1579     510
1580     511
1581     512
1582     513
1583     514
1584     515
1585     516
1586     517
1587     518
1588     519
1589     520
1590     521
1591     522
1592     523
1593     524
1594     525
1595     526
1596     527
1597     528
1598     529
1599     530
1600     531
1601     532
1602     533
1603     534
1604     535
1605     536
1606     537
1607     538
1608     539
1609     540
1610     541
1611     542
1612     543
1613     544
1614     545
1615     546
1616     547
1617     548
1618     549
1619     550
1620     551
1621     552
1622     553
1623     554
1624     555
1625     556
1626     557
1627     558
1628     559
1629     560
1630     561
1631     562
1632     563
1633     564
1634     565
1635     566
1636     567
1637     568
1638     569
1639     570
1640     571
1641     572
1642     573
1643     574
1644     575
1645     576
1646     577
1647     578
1648     579
1649     580
1650     581
1651     582
1652     583
1653     584
1654     585
1655     586
1656     587
1657     588
1658     589
1659     590
1660     591
1661     592
1662     593
1663     594
1664     595
1665     596
1666     597
1667     598
1668     599
1669     600
1670     601
1671     602
1672     603
1673     604
1674     605
1675     606
1676     607
1677     608
1678     609
1679     610
1680     611
1681     612
1682     613
1683     614
1684     615
1685     616
1686     617
1687     618
1688     619
1689     620
1690     621
1691     622
1692     623
1693     624
1694     625
1695     626
1696     627
1697     628
1698     629
1699     630
1700     631
1701     632
1702     633
1703     634
1704     635
1705     636
1706     637
1707     638
1708     639
1709     640
1710     641
1711     642
1712     643
1713     644
1714     645
1715     646
1716     647
1717     648
1718     649
1719     650
1720     651
1721     652
1722     653
1723     654
1724     655
1725     656
1726     657
1727     658
1728     659
1729     660
1730     661
1731     662
1732     663
1733     664
1734     665
1735     666
1736     667
1737     668
1738     669
1739     670
1740     671
1741     672
1742     673
1743     674
1744     675
1745     676
1746     677
1747     678
1748     679
1749     680
1750     681
1751     682
1752     683
1753     684
1754     685
1755     686
1756     687
1757     688
1758     689
1759     690
1760     691
1761     692
1762     693
1763     694
1764     695
1765     696
1766     697
1767     698
1768     699
1769     700
1770     701
1771     702
1772     703
1773     704
1774     705
1775     706
1776     707
1777     708
1778     709
1779     710
1780     711
1781     712
1782     713
1783     714
1784     715
1785     716
1786     717
1787     718
1788     719
1789     720
1790     721
1791     722
1792     723
1793     724
1794     725
1795     726
1796     727
1797     728
1798     729
1799     730
1800     731
1801     732
1802     733
1803     734
1804     735
1805     736
1806     737
1807     738
1808     739
1809     740
1810     741
1811     742
1812     743
1813     744
1814     745
1815     746
1816     747
1817     748
1818     749
1819     750
1820     751
1821     752
1822     753
1823     754
1824     755
1825     756
1826     757
1827     758
1828     759
1829     760
1830     761
1831     762
1832     763
1833     764
1834     765
1835     766
1836     767
1837     768
1838     769
1839     770
1840     771
1841     772
1842     773
1843     774
1844     775
1845     776
1846     777
1847     778
1848     779
1849     780
1850     781
1851     782
1852     783
1853     784
1854     785
1855     786
1856     787
1857     788
1858     789
1859     790
1860     791
1861     792
1862     793
1863     794
1864     795
1865     796
1866     797
1867     798
1868     799
1869     800
1870     801
1871     802
1872     803
1873     804
1874     805
1875     806
1876     807
1877     808
1878     809
1879     810
1880     811
1881     812
1882     813
1883     814
1884     815
1885     816
1886     817
1887     818
1888     819
1889     820
1890     821
1891     822
1892     823
1893     824
1894     825
1895     826
1896     827
1897     828
1898     829
1899     830
1900     831
1901     832
1902     833
1903     834
1904     835
1905     836
1906     837
1907     838
1908     839
1909     840
1910     841
1911     842
1912     843
1913     844
1914     845
1915     846
1916     847
1917     848
1918     849
1919     850
1920     851
1921     852
1922     853
1923     854
1924     855
1925     856
1926     857
1927     858
1928     859
1929     860
1930     861
1931     862
1932     863
1933     864
1934     865
1935     866
1936     867
1937     868
1938     869
1939     870
1940     871
1941     872
1942     873
1943     874
1944     875
1945     876
1946     877
1947     878
1948     879
1949     880
1950     881
1951     882
1952     883
1953     884
1954     885
1955     886
1956     887
1957     888
1958     889
1959     890
1960     891
1961     892
1962     893
1963     894
1964     895
1965     896
1966     897
1967     898
1968     899
1969     900
1970     901
1971     902
1972     903
1973     904
1974     905
1975     906
1976     907
1977     908
1978     909
1979     910
1980     911
1981     912
1982     913
1983     914
1984     915
1985     916
1986     917
1987     918
1988     919
1989     920
1990     921
1991     922
1992     923
1993     924
1994     925
1995     926
1996     927
1997     928
1998     929
1999     930
2000     931
2001     932
2002     933
2003     934
2004     935
2005     936
2006     937
2007     938
2008     939
2009     940
2010     941
2011     942
2012     943
2013     944
2014     945
2015     946
2016     947
2017     948
2018     949
2019     950
2020     951
2021     952
2022     953
2023     954
2024     955
2025     956
2026     957
2027     958
2028     959
2029     960
2030     961
2031     962
2032     963
2033     964
2034     965
2035     966
2036     967
2037     968
2038     969
2039     970
2040     971
2041     972
2042     973
2043     974
2044     975
2045     976
2046     977
2047     978
2048     979
2049     980
2050     981
2051     982
2052     983
2053     984
2054     985
2055     986
2056     987
2057     988
2058     989
2059     990
2060     991
2061     992
2062     993
2063     994
2064     995
2065     996
2066     997
2067     998
2068     999
2069     1000
2070     1001
2071     1002
2072     1003
2073     1004
2074     1005
2075     1006
2076     1007
2077     1008
2078     1009
2079     1010
2080     1011
2081     1012
2082     1013
2083     1014
2084     1015
2085     1016
2086     1017
2087     1018
2088     1019
2089     1020
2090     1021
2091     1022
2092     1023
2093     1024
2094     1025
2095     1026
2096     1027
2097     1028
2098     1029
2099     1030
2100     1031
2101     1032
2102     1033
2103     1034
2104     1035
2105     1036
2106     1037
2107     1038
2108     1039
2109     1040
2110     1041
2111     1042
2112     1043
2113     1044
2114     1045
2115     1046
2116     1047
2117     1048
2118     1049
2119     1050
2120     1051
2121     1052
2122     1053
2123     1054
2124     1055
2125     1056
2126     1057
2127     1058
2128     1059
2129     1060
2130     1061
2131     1062
2132     1063
2133     1064
2134     1065
2135     1066
2136     1067
2137     1068
2138     1069
2139     1070
2140     1071
2141     1072
2142     1073
2143     1074
2144     1075
2145     1076
2146     1077
2147     1078
2148     1079
2149     1080
2150     1081
2151     1082
2152     1083
2153     1084
2154     1085
2155     1086
2156     1087
2157     1088
2158     1089
2159     1090
2160     1091
2161     1092
2162     1093
2163     1094
2164     1095
2165     1096
2166     1097
2167     1098
2168     1099
2169     1100
2170     1101
2171     1102
2172     1103
2173     1104
2174     1105
2175     1106
2176     1107
2177     1108
2178     1109
2179     1110
2180     1111
2181     1112
2182     1113
2183     1114
2184     1115
2
```

```

113419     As an executor, you must:
113520     - Explicitly verify all conditions
113621     - Show complete mathematical reasoning
113722     - Include standalone line: "Final Answer: \boxed{{answer}}"
113823     - Ensure your answer directly responds to the question asked
113924     - The final answer should be one exact number
114025     - Not all conditions can serve as the conditions for solving
114126         problems. We should answer according to the problems
114227 """
114328     response = await self.llm.generate(prompt)
114429     answer = self._extract_answer(response)
114530     if answer:
114631         node.answer = answer
114732         node.state = "solved"
114833     return {
114934         "node_id": node_id,
115035         "response": response,
115136         "answer": answer
115237     }
115338     return None
115439
115540     async def _aggregation(self, solutions: List[Dict[str, Any]]) -> str:
115641         if not solutions:
115742             return "No valid solutions found"
115843         original_question = None
115944         for sol in solutions:
116045             node = self.nodes[sol["node_id"]]
116146             if hasattr(node, 'original_question'):
116247                 original_question = node.original_question
116348                 break
116449         if original_question is None:
116550             first_node = self.nodes[solutions[0]["node_id"]]
116651             path = first_node.path
116752             if path:
116853                 root_node_id = path[0]
116954                 root_node = self.nodes.get(root_node_id)
117055                 if root_node:
117156                     original_question = root_node.method.get("description", "Original problem")
117257         if original_question is None:
117358             original_question = "Original problem (reconstructed from context"
117459             ")
117560         if solutions[0]["response"]:
117661             match = re.search(r'Original Problem[:\s]*(.+?)\nSteps:',
117762                         solutions[0]["response"])
117863             if match:
117964                 original_question = match.group(1).strip()
118065         if len(solutions) == 1:
118166             return solutions[0]["answer"]
118267         unique_answers = {sol["answer"] for sol in solutions}
118368         if len(unique_answers) == 1:
118469             return solutions[0]["answer"]
118570         solutions_text = "\n\n".join(
118671             f"Solution {i+1} (Node: {sol['node_id']}):\n"
118772             f"Answer: {sol['answer']}\n"
118873             f"Approach: {self.nodes[sol['node_id']].method['description']}\n"
118974             f"conditions: {self.nodes[sol['node_id']].conditions}\n"
119075             f"Reasoning Excerpt:\n{solutions[0]['response'][:300]}...\n"
119176             for i, sol in enumerate(solutions)
119277         )
119378         prompt = f"""
119479             You are a world-class mathematician and mathematical logician.
119580             You are intelligent, rigorous, and cautious.
119681             You always reason step by step, consider all relevant conditions.

```

```

1188      You think in terms of structure, symmetry, and mathematical
1189      principles, and never skip important logical steps.
1190      You aim to find a complete and correct solution, not just an
1191      answer.
1192      You THINK CLEARLY, STRUCTURALLY, AND DEEPLY.
1193      Synthesize these solutions for the original problem:
1194      Original Problem: {original_question}
1195      Proposed Solutions:
1196      {solutions_text}
1197      As an analyst, you must:
1198      1. FIRST verify which solution(s) correctly answer the original
         question
1199      2. Compare mathematical consistency with the original problem
         statement
1200      3. Evaluate which approach best satisfies all conditions
1201      4. Combine elements from multiple solutions ONLY if
         mathematically valid
1202      5. Provide clear justification for your selection
1203      6. Mark final answer with \boxed{{}}
1204      7. Include standalone line: "Aggregated Answer: answer"
1205      Critical Analysis Guidelines:
1206      - The solution MUST directly answer the original question as
         stated
1207      - Prioritize mathematical correctness over elegance
1208      - Reject solutions that violate any explicit conditions
1209      - Verify all intermediate calculations are sound
1210      - Ensure the final answer format matches what the problem
         requires
1211      """
1212      response = await self.llm.generate(prompt)
1213      return self._extract_answer(response) or "Aggregation failed"
1214

```

Listing 4: Subquestion Resolution and Aggregation Prompt in Mathematics

A.5 PROMPTS OF **HoT** IN LOGIC

In this section, we introduce prompts for each module of **HoT** in Logic (OpenBookQA).

A.5.1 CONDITION EXTRACTION

```

1  async def _condition_extraction(self, question: str, options: Dict[str,
2      str]) -> Dict[str, Any]:
3      prompt = f"""
4          You are a top expert in formal logic, critical thinking, and
5          argument analysis.
6          You are precise, rational, and skeptical.
7          You always examine each statement carefully, identify premises
8          and conclusions, and evaluate logical validity step by step.
9          You avoid unwarranted assumptions, think in terms of logical
10         consequences, and eliminate invalid options with sound
11         reasoning.
12         You aim to reach conclusions based only on evidence and logic.
13         You THINK SLOWLY, CAREFULLY, AND LOGICALLY.
14         Analyze this question and extract key conditions:
15
16         Question: {question}
17         Options:
18         A. {options['A']}
19         B. {options['B']}
20         C. {options['C']}
21         D. {options['D']}
22
23         Identify:
24         1. Explicit conditions (directly stated)

```

```
1242 20
1243 21
1244 22
1245 23
1246 24
1247 25
1248 26
1249 27
1250 28
1251 29
1252 30
1253 31
1254 32
1255 33
1256 34
1257 35
1258 36
1259 37
1260 38
1261 39
1262 40
1263 41
1264 42
1265 43
1266 44
    2. Implicit conditions (logical implications)
    3. Key terms and their relationships
    4. Spatial/temporal relationships if present
    5. Any conditional statements

    Output JSON format:
    {{ "explicit": ["list", "of", "conditions"], "implicit": ["list", "of", "conditions"], "key_terms": ["term1", "term2"], "notes": "Analysis summary" } }

    """
    for attempt in range(self.config.max_retries):
        try:
            response = await self.llm.generate(prompt, response_format="json_object")
            return json.loads(response)
        except:
            continue
    return {
        "explicit": [],
        "implicit": [],
        "key_terms": [],
        "notes": "Failed to extract conditions"
    }
```

Listing 5: Condition Extraction Prompt in Logic

A.5.2 TREE EXPLORER

```

1296                 "score_reason": "Scoring justification"
1297             }
1298         ]
1299     }
1300 """
1301 for attempt in range(self.config.max_retries):
1302     try:
1303         response = await self.llm.generate(prompt, response_format=""
1304                                         json_object")
1305         response = response.strip()
1306         if response.startswith("```json"):
1307             response = response[7:-3].strip()
1308         elif response.startswith("```"):
1309             response = response[3:-3].strip()
1310         data = json.loads(response)
1311         if not isinstance(data, dict) or "methods" not in data:
1312             raise ValueError("Invalid structure: missing 'methods'
1313                             key")
1314         methods = data["methods"]
1315         if len(methods) < 2:
1316             raise ValueError(f"Expected at least 2 methods, got {len(
1317                                         methods)}")
1318         required_keys = {"description", "steps", "score", ""
1319                         score_reason"}
1320         for method in methods:
1321             if not all(k in method for k in required_keys):
1322                 raise ValueError("Missing required keys in method")
1323             if not isinstance(method["steps"], list):
1324                 raise ValueError("Steps must be a list")
1325         return sorted(methods, key=lambda x: -x["score"])
1326     except (json.JSONDecodeError, ValueError, KeyError) as e:
1327         if attempt == self.config.max_retries - 1:
1328             return []
1329         continue
1330     return []

```

Listing 6: Tree Explorer Prompt in Logic

A.5.3 ADAPTIVE DOMAIN DECOMPOSITION

```

1330
1331     1 async def _adaptive_domain_decomposition(self, method: str, question: str
1332     , options: Dict[str, str]) -> Dict[str, Any]:
1333     2     options_text = "\n".join([f"\n{k}. {v}" for k, v in options.items()])
1334     3     prompt = f"""
1335     4         You are a top expert in formal logic, critical thinking, and
1336         argument analysis.
1337     5         You are precise, rational, and skeptical.
1338     6         You always examine each statement carefully, identify premises
1339         and conclusions, and evaluate logical validity step by step.
1340     7         You avoid unwarranted assumptions, think in terms of logical
1341         consequences, and eliminate invalid options with sound
1342         reasoning.
1343     8         You aim to reach conclusions based only on evidence and logic.
1344     9         You THINK SLOWLY, CAREFULLY, AND LOGICALLY.
1345    10         Determine if this solution approach requires case classification:
1346    11         Solution Approach: {method}
1347    12         Question: {question}
1348    13         Options:
1349    14             {options_text}
1350    15             Consider:
1351    16                 1. Does the question contain multiple scenarios or cases?
1352    17                 2. Are there conditional statements that create distinct
1353                     possibilities?
1354    18                 3. Do the options represent different logical paths?

```

```

1350
1351         4. Would different initial assumptions lead to different
1352             solutions?
1353             If classification needed, provide:
1354             - Comprehensive case descriptions
1355             - Precise conditions for each case
1356             - Expected outcomes
1357             Output JSON format:
1358             {{{
1359                 "need_classify": true/false,
1360                 "reason": "Classification rationale",
1361                 "cases": [
1362                     {{{
1363                         "description": "Case description",
1364                         "conditions": {"parameter": "value_range"} }}}
1365                     ] }
1366                 } }
1367             """
1368         try:
1369             response = await self.llm.generate(prompt, response_format=""
1370                 json_object)
1371             data = json.loads(response)
1372             return data
1373         except:
1374             return {
1375                 "need_classify": False,
1376                 "reason": "Analysis failed",
1377                 "cases": []
1378             }

```

Listing 7: Adaptive Domain Decomposition Prompt in Logic

1378 A.5.4 SUBQUESTION RESOLUTION AND AGGREGATION

```

1379
1380     1 async def _resolution(self, node_id: str) -> Optional[Dict[str, Any]]:
1381         2     node = self.nodes[node_id]
1382         3     context = f"Question: {node.question}\nOptions:\n"
1383         4     for opt, text in node.options.items():
1384         5         context += f"{opt}. {text}\n"
1385         6     context += f"\nSolution Approach: {node.method['description']}\n"
1386         7     context += f"conditions: {json.dumps(node.conditions, indent=2)}\n"
1387         8     prompt = f"""
1388             9         You are a top expert in formal logic, critical thinking, and
1389             10            argument analysis.
1390             11            You are precise, rational, and skeptical.
1391             12            You always examine each statement carefully, identify premises
1392             13            and conclusions, and evaluate logical validity step by step.
1393             14            You avoid unwarranted assumptions, think in terms of logical
1394             15            consequences, and eliminate invalid options with sound
1395             16            reasoning.
1396             17            You aim to reach conclusions based only on evidence and logic.
1397             18            You THINK SLOWLY, CAREFULLY, AND LOGICALLY.
1398             19            Solve this question using the specified approach:
1399             20            {context}
1400             21            Reasoning Steps:
1401             22            1. Strictly follow the provided approach: {node.method['
1402             23            description']}
1403             24            2. Execute each step: {', '.join(node.method['steps'])}
1404             25            3. Consider all conditions
1405             26            4. Evaluate each option systematically
1406             27            5. Provide clear justification for inclusion/exclusion
1407             28            6. Select the best answer
1408             29            Output Requirements:
1409             30            - End your response with: "Final Answer: [OPTION]"

```

```

1404 26     - Use \boxed{{[OPTION]}} to denote your answer
1405 27     - Your answer must be A, B, C, or D
1406 28 """
1407 29     response = await self.llm.generate(prompt)
1408 30     answer = self._extract_answer(response)
1409 31     if answer:
1410 32         node.answer = answer
1411 33         node.state = "solved"
1412 34         return {
1413 35             "node_id": node_id,
1414 36             "response": response,
1415 37             "answer": answer
1416 38         }
1417 39     return None
1418 40
1419 41     async def _aggregation(self, solutions: List[Dict[str, Any]]) -> str:
1420 42         if not solutions:
1421 43             return "X" # Invalid answer
1422 44         if len(solutions) == 1:
1423 45             return solutions[0]["answer"]
1424 46         answers = [s["answer"] for s in solutions]
1425 47         if len(set(answers)) == 1:
1426 48             return answers[0]
1427 49         solutions_text = ""
1428 50         for i, sol in enumerate(solutions):
1429 51             node = self.nodes[sol["node_id"]]
1430 52             solutions_text += f"\n\nSolution {i+1} (Node {sol['node_id']}):"
1431 53             solutions_text += f"\nApproach: {node.method['description']}"
1432 54             solutions_text += f"\nconditions: {json.dumps(node.conditions,
1433 55                 indent=2)}"
1434 56             solutions_text += f"\nAnswer: {sol['answer']}"
1435 57             solutions_text += f"\nReasoning Excerpt:\n{sol['response']}..."
1436 58
1437 59     prompt = f"""
1438 60         You are a top expert in formal logic, critical thinking, and
1439 61             argument analysis.
1440 62         You are precise, rational, and skeptical.
1441 63         You always examine each statement carefully, identify premises
1442 64             and conclusions, and evaluate logical validity step by step.
1443 65         You avoid unwarranted assumptions, think in terms of logical
1444 66             consequences, and eliminate invalid options with sound
1445 67             reasoning.
1446 68         You aim to reach conclusions based only on evidence and logic.
1447 69         You THINK SLOWLY, CAREFULLY, AND LOGICALLY.
1448 70         Synthesize these approaches:
1449 71
1450 72             {solutions_text}
1451 73
1452 74             Instructions:
1453 75                 1. Analyze all solutions and their approaches
1454 76                 2. Identify the most reliable reasoning
1455 77                 3. Verify consistency with conditions
1456 78                 4. Select the best overall answer
1457 79                 5. Output format: \boxed{{[ANSWER]}}
1458 80 """
1459 81     response = await self.llm.generate(prompt)
1460 82     return self._extract_answer(response) or "X"
1461 83
1462 84
1463 85
1464 86
1465 87
1466 88
1467 89
1468 90
1469 91
1470 92
1471 93
1472 94
1473 95
1474 96
1475 97
1476 98
1477 99
1478 100
1479 101
1480 102
1481 103
1482 104
1483 105
1484 106
1485 107
1486 108
1487 109
1488 110
1489 111
1490 112
1491 113
1492 114
1493 115
1494 116
1495 117
1496 118
1497 119
1498 120
1499 121
1500 122
1501 123
1502 124
1503 125
1504 126
1505 127
1506 128
1507 129
1508 130
1509 131
1510 132
1511 133
1512 134
1513 135
1514 136
1515 137
1516 138
1517 139
1518 140
1519 141
1520 142
1521 143
1522 144
1523 145
1524 146
1525 147
1526 148
1527 149
1528 150
1529 151
1530 152
1531 153
1532 154
1533 155
1534 156
1535 157
1536 158
1537 159
1538 160
1539 161
1540 162
1541 163
1542 164
1543 165
1544 166
1545 167
1546 168
1547 169
1548 170
1549 171
1550 172
1551 173
1552 174
1553 175
1554 176
1555 177
1556 178
1557 179
1558 180
1559 181
1560 182
1561 183
1562 184
1563 185
1564 186
1565 187
1566 188
1567 189
1568 190
1569 191
1570 192
1571 193
1572 194
1573 195
1574 196
1575 197
1576 198
1577 199
1578 200
1579 201
1580 202
1581 203
1582 204
1583 205
1584 206
1585 207
1586 208
1587 209
1588 210
1589 211
1590 212
1591 213
1592 214
1593 215
1594 216
1595 217
1596 218
1597 219
1598 220
1599 221
1600 222
1601 223
1602 224
1603 225
1604 226
1605 227
1606 228
1607 229
1608 230
1609 231
1610 232
1611 233
1612 234
1613 235
1614 236
1615 237
1616 238
1617 239
1618 240
1619 241
1620 242
1621 243
1622 244
1623 245
1624 246
1625 247
1626 248
1627 249
1628 250
1629 251
1630 252
1631 253
1632 254
1633 255
1634 256
1635 257
1636 258
1637 259
1638 260
1639 261
1640 262
1641 263
1642 264
1643 265
1644 266
1645 267
1646 268
1647 269
1648 270
1649 271
1650 272
1651 273
1652 274
1653 275
1654 276
1655 277
1656 278
1657 279
1658 280
1659 281
1660 282
1661 283
1662 284
1663 285
1664 286
1665 287
1666 288
1667 289
1668 290
1669 291
1670 292
1671 293
1672 294
1673 295
1674 296
1675 297
1676 298
1677 299
1678 300
1679 301
1680 302
1681 303
1682 304
1683 305
1684 306
1685 307
1686 308
1687 309
1688 310
1689 311
1690 312
1691 313
1692 314
1693 315
1694 316
1695 317
1696 318
1697 319
1698 320
1699 321
1700 322
1701 323
1702 324
1703 325
1704 326
1705 327
1706 328
1707 329
1708 330
1709 331
1710 332
1711 333
1712 334
1713 335
1714 336
1715 337
1716 338
1717 339
1718 340
1719 341
1720 342
1721 343
1722 344
1723 345
1724 346
1725 347
1726 348
1727 349
1728 350
1729 351
1730 352
1731 353
1732 354
1733 355
1734 356
1735 357
1736 358
1737 359
1738 360
1739 361
1740 362
1741 363
1742 364
1743 365
1744 366
1745 367
1746 368
1747 369
1748 370
1749 371
1750 372
1751 373
1752 374
1753 375
1754 376
1755 377
1756 378
1757 379
1758 380
1759 381
1760 382
1761 383
1762 384
1763 385
1764 386
1765 387
1766 388
1767 389
1768 390
1769 391
1770 392
1771 393
1772 394
1773 395
1774 396
1775 397
1776 398
1777 399
1778 400
1779 401
1780 402
1781 403
1782 404
1783 405
1784 406
1785 407
1786 408
1787 409
1788 410
1789 411
1790 412
1791 413
1792 414
1793 415
1794 416
1795 417
1796 418
1797 419
1798 420
1799 421
1800 422
1801 423
1802 424
1803 425
1804 426
1805 427
1806 428
1807 429
1808 430
1809 431
1810 432
1811 433
1812 434
1813 435
1814 436
1815 437
1816 438
1817 439
1818 440
1819 441
1820 442
1821 443
1822 444
1823 445
1824 446
1825 447
1826 448
1827 449
1828 450
1829 451
1830 452
1831 453
1832 454
1833 455
1834 456
1835 457
1836 458
1837 459
1838 460
1839 461
1840 462
1841 463
1842 464
1843 465
1844 466
1845 467
1846 468
1847 469
1848 470
1849 471
1850 472
1851 473
1852 474
1853 475
1854 476
1855 477
1856 478
1857 479
1858 480
1859 481
1860 482
1861 483
1862 484
1863 485
1864 486
1865 487
1866 488
1867 489
1868 490
1869 491
1870 492
1871 493
1872 494
1873 495
1874 496
1875 497
1876 498
1877 499
1878 500
1879 501
1880 502
1881 503
1882 504
1883 505
1884 506
1885 507
1886 508
1887 509
1888 510
1889 511
1890 512
1891 513
1892 514
1893 515
1894 516
1895 517
1896 518
1897 519
1898 520
1899 521
1900 522
1901 523
1902 524
1903 525
1904 526
1905 527
1906 528
1907 529
1908 530
1909 531
1910 532
1911 533
1912 534
1913 535
1914 536
1915 537
1916 538
1917 539
1918 540
1919 541
1920 542
1921 543
1922 544
1923 545
1924 546
1925 547
1926 548
1927 549
1928 550
1929 551
1930 552
1931 553
1932 554
1933 555
1934 556
1935 557
1936 558
1937 559
1938 560
1939 561
1940 562
1941 563
1942 564
1943 565
1944 566
1945 567
1946 568
1947 569
1948 570
1949 571
1950 572
1951 573
1952 574
1953 575
1954 576
1955 577
1956 578
1957 579
1958 580
1959 581
1960 582
1961 583
1962 584
1963 585
1964 586
1965 587
1966 588
1967 589
1968 590
1969 591
1970 592
1971 593
1972 594
1973 595
1974 596
1975 597
1976 598
1977 599
1978 600
1979 601
1980 602
1981 603
1982 604
1983 605
1984 606
1985 607
1986 608
1987 609
1988 610
1989 611
1990 612
1991 613
1992 614
1993 615
1994 616
1995 617
1996 618
1997 619
1998 620
1999 621
2000 622
2001 623
2002 624
2003 625
2004 626
2005 627
2006 628
2007 629
2008 630
2009 631
2010 632
2011 633
2012 634
2013 635
2014 636
2015 637
2016 638
2017 639
2018 640
2019 641
2020 642
2021 643
2022 644
2023 645
2024 646
2025 647
2026 648
2027 649
2028 650
2029 651
2030 652
2031 653
2032 654
2033 655
2034 656
2035 657
2036 658
2037 659
2038 660
2039 661
2040 662
2041 663
2042 664
2043 665
2044 666
2045 667
2046 668
2047 669
2048 670
2049 671
2050 672
2051 673
2052 674
2053 675
2054 676
2055 677
2056 678
2057 679
2058 680
2059 681
2060 682
2061 683
2062 684
2063 685
2064 686
2065 687
2066 688
2067 689
2068 690
2069 691
2070 692
2071 693
2072 694
2073 695
2074 696
2075 697
2076 698
2077 699
2078 700
2079 701
2080 702
2081 703
2082 704
2083 705
2084 706
2085 707
2086 708
2087 709
2088 710
2089 711
2090 712
2091 713
2092 714
2093 715
2094 716
2095 717
2096 718
2097 719
2098 720
2099 721
2100 722
2101 723
2102 724
2103 725
2104 726
2105 727
2106 728
2107 729
2108 730
2109 731
2110 732
2111 733
2112 734
2113 735
2114 736
2115 737
2116 738
2117 739
2118 740
2119 741
2120 742
2121 743
2122 744
2123 745
2124 746
2125 747
2126 748
2127 749
2128 750
2129 751
2130 752
2131 753
2132 754
2133 755
2134 756
2135 757
2136 758
2137 759
2138 760
2139 761
2140 762
2141 763
2142 764
2143 765
2144 766
2145 767
2146 768
2147 769
2148 770
2149 771
2150 772
2151 773
2152 774
2153 775
2154 776
2155 777
2156 778
2157 779
2158 780
2159 781
2160 782
2161 783
2162 784
2163 785
2164 786
2165 787
2166 788
2167 789
2168 790
2169 791
2170 792
2171 793
2172 794
2173 795
2174 796
2175 797
2176 798
2177 799
2178 800
2179 801
2180 802
2181 803
2182 804
2183 805
2184 806
2185 807
2186 808
2187 809
2188 810
2189 811
2190 812
2191 813
2192 814
2193 815
2194 816
2195 817
2196 818
2197 819
2198 820
2199 821
2200 822
2201 823
2202 824
2203 825
2204 826
2205 827
2206 828
2207 829
2208 830
2209 831
2210 832
2211 833
2212 834
2213 835
2214 836
2215 837
2216 838
2217 839
2218 840
2219 841
2220 842
2221 843
2222 844
2223 845
2224 846
2225 847
2226 848
2227 849
2228 850
2229 851
2230 852
2231 853
2232 854
2233 855
2234 856
2235 857
2236 858
2237 859
2238 860
2239 861
2240 862
2241 863
2242 864
2243 865
2244 866
2245 867
2246 868
2247 869
2248 870
2249 871
2250 872
2251 873
2252 874
2253 875
2254 876
2255 877
2256 878
2257 879
2258 880
2259 881
2260 882
2261 883
2262 884
2263 885
2264 886
2265 887
2266 888
2267 889
2268 890
2269 891
2270 892
2271 893
2272 894
2273 895
2274 896
2275 897
2276 898
2277 899
2278 900
2279 901
2280 902
2281 903
2282 904
2283 905
2284 906
2285 907
2286 908
2287 909
2288 910
2289 911
2290 912
2291 913
2292 914
2293 915
2294 916
2295 917
2296 918
2297 919
2298 920
2299 921
2300 922
2301 923
2302 924
2303 925
2304 926
2305 927
2306 928
2307 929
2308 930
2309 931
2310 932
2311 933
2312 934
2313 935
2314 936
2315 937
2316 938
2317 939
2318 940
2319 941
2320 942
2321 943
2322 944
2323 945
2324 946
2325 947
2326 948
2327 949
2328 950
2329 951
2330 952
2331 953
2332 954
2333 955
2334 956
2335 957
2336 958
2337 959
2338 960
2339 961
2340 962
2341 963
2342 964
2343 965
2344 966
2345 967
2346 968
2347 969
2348 970
2349 971
2350 972
2351 973
2352 974
2353 975
2354 976
2355 977
2356 978
2357 979
2358 980
2359 981
2360 982
2361 983
2362 984
2363 985
2364 986
2365 987
2366 988
2367 989
2368 990
2369 991
2370 992
2371 993
2372 994
2373 995
2374 996
2375 997
2376 998
2377 999
2378 999
2379 999
2380 999
2381 999
2382 999
2383 999
2384 999
2385 999
2386 999
2387 999
2388 999
2389 999
2390 999
2391 999
2392 999
2393 999
2394 999
2395 999
2396 999
2397 999
2398 999
2399 999
2400 999
2401 999
2402 999
2403 999
2404 999
2405 999
2406 999
2407 999
2408 999
2409 999
2410 999
2411 999
2412 999
2413 999
2414 999
2415 999
2416 999
2417 999
2418 999
2419 999
2420 999
2421 999
2422 999
2423 999
2424 999
2425 999
2426 999
2427 999
2428 999
2429 999
2430 999
2431 999
2432 999
2433 999
2434 999
2435 999
2436 999
2437 999
2438 999
2439 999
2440 999
2441 999
2442 999
2443 999
2444 999
2445 999
2446 999
2447 999
2448 999
2449 999
2450 999
2451 999
2452 999
2453 999
2454 999
2455 999
2456 999
2457 999
2458 999
2459 999
2460 999
2461 999
2462 999
2463 999
2464 999
2465 999
2466 999
2467 999
2468 999
2469 999
2470 999
2471 999
2472 999
2473 999
2474 999
2475 999
2476 999
2477 999
2478 999
2479 999
2480 999
2481 999
2482 999
2483 999
2484 999
2485 999
2486 999
2487 999
2488 999
2489 999
2490 999
2491 999
2492 999
2493 999
2494 999
2495 999
2496 999
2497 999
2498 999
2499 999
2500 999
2501 999
2502 999
2503 999
2504 999
2505 999
2506 999
2507 999
2508 999
2509 999
2510 999
2511 999
2512 999
2513 999
2514 999
2515 999
2516 999
2517 999
2518 999
2519 999
2520 999
2521 999
2522 999
2523 999
2524 999
2525 999
2526 999
2527 999
2528 999
2529 999
2530 999
2531 999
2532 999
2533 999
2534 999
2535 999
2536 999
2537 999
2538 999
2539 999
2540 999
2541 999
2542 999
2543 999
2544 999
2545 999
2546 999
2547 999
2548 999
2549 999
2550 999
2551 999
2552 999
2553 999
2554 999
2555 999
2556 999
2557 999
2558 999
2559 999
2560 999
2561 999
2562 999
2563 999
2564 999
2565 999
2566 999
2567 999
2568 999
2569 999
2570 999
2571 999
2572 999
2573 999
2574 999
2575 999
2576 999
2577 999
2578 999
2579 999
2580 999
2581 999
2582 999
2583 999
2584 999
2585 999
2586 999
2587 999
2588 999
2589 999
2590 999
2591 999
2592 999
2593 999
2594 999
2595 999
2596 999
2597 999
2598 999
2599 999
2600 999
2601 999
2602 999
2603 999
2604 999
2605 999
2606 999
2607 999
2608 999
2609 999
2610 999
2611 999
2612 999
2613 999
2614 999
2615 999
2616 999
2617 999
2618 999
2619 999
2620 999
2621 999
2622 999
2623 999
2624 999
2625 999
2626 999
2627 999
2628 999
2629 999
2630 999
2631 999
2632 999
2633 999
2634 999
2635 999
2636 999
2637 999
2638 999
2639 999
2640 999
2641 999
2642 999
2643 999
2644 999
2645 999
2646 999
2647 999
2648 999
2649 999
2650 999
2651 999
2652 999
2653 999
2654 999
2655 999
2656 999
2657 999
2658 999
2659 999
2660 999
2661 999
2662 999
2663 999
2664 999
2665 999
2666 999
2667 999
2668 999
2669 999
2670 999
2671 999
2672 999
2673 999
2674 999
2675 999
2676 999
2677 999
2678 999
2679 999
2680 999
2681 999
2682 999
2683 999
2684 999
2685 999
2686 999
2687 999
2688 999
2689 999
2690 999
2691 999
2692 999
2693 999
2694 999
2695 999
2696 999
2697 999
2698 999
2699 999
2700 999
2701 999
2702 999
2703 999
2704 999
2705 999
2706 999
2707 999
2708 999
2709 999
2710 999
2711 999
2712 999
2713 999
2714 999
2715 999
2716 999
2717 999
2718 999
2719 999
2720 999
2721 999
2722 999
2723 999
2724 999
2725 999
2726 999
2727 999
2728 999
2729 999
2730 999
2731 999
2732 999
2733 999
273
```

Listing 8: Subquestion Resolution and Aggregation Prompt in Logic

1458 A.6 IMPLEMENTATION DETAILS

1459

1460 A.6.1 REACT

1461

1462 We adopt a *few-shot ReAct-style* prompting strategy inspired by the ReAct framework. While the
 1463 original ReAct framework combines language model reasoning with external tool-use, our imple-
 1464 mentation simplifies this structure by eliminating actual API calls or tool integration. Instead, we
 1465 simulate both reasoning (Thought) and acting (Action) steps purely within natural language,
 1466 forming a lightweight and deployable version suitable for standard API-based model access.

1467 **Comparison with Original ReAct** The original ReAct framework relies on dynamic tool use:
 1468 the model emits an *Action*, receives an *Observation*, and continues reasoning based on this
 1469 feedback loop. This mechanism enhances performance on tasks requiring retrieval or real-time
 1470 computation.

1471

1472 In contrast, our ReAct-style prompting:

1473

- Requires no tool infrastructure or external observation integration.
- Can be used directly with closed-source APIs, enabling plug-and-play reasoning for both mathematical and commonsense questions.
- Emphasizes interpretability through explicit intermediate reasoning chains and simulated actions.

1474

1475 Thus, our method trades the dynamism of tool interaction for broad compatibility and simplicity,
 1476 enabling structured reasoning with minimal implementation overhead. This makes it well-suited as
 1477 a practical baseline for both mathematical and logical benchmarks.

1478

1479 Original Format:

1480

1481 **Question:** The natural language query to be solved.

1482

1483 **Thought:** Intermediate steps expressed as natural language reasoning.

1484

1485 **Action (optional):** If needed, simulate tool-use as internal calculation.

1486

1487 **Final Answer:** Explicit, boxed or stated final response.

1488

1489

1490 Example 1 (GSM8K-style Arithmetic Reasoning):

1491

1492 **Question:** Farmer Brown has 20 animals on his farm, all either chickens or cows.
 1493 They have a total of 70 legs altogether. How many of the animals are chickens?

1494

1495 **Thought:** Let C be the number of chickens. Then the number of cows is $20 - C$.
 Chickens have 2 legs, cows have 4. So total legs = $2C + 4(20 - C) = 70$.

1496

1497 **Thought:** Simplify the equation: $2C + 80 - 4C = 70 \Rightarrow -2C = -10 \Rightarrow C = 5$.

1498

1499 **Final Answer:** 5

1500

1501 Example 2 (OpenBookQA-style Commonsense Reasoning)

1502

1503 **Question:** Some animals use a liquid coming from their skin to adjust to

1504

1505 **Choices:** A) cold B) water C) heat D) humidity

1506

1507 **Thought:** Many animals sweat to regulate their body temperature when it is hot.
 1508 Sweat is a liquid that comes from the skin.

1509

1510 **Final Answer:** C

1511

1512 A.6.2 INSTANCE-ADAPTIVE PROMPTING (IAP)

1513

1514 We implement the IAP strategy with two variants: Majority Vote (IAP-mv) and Sequential Substi-
 1515 tution (IAP-ss). The implementation builds upon the Qwen2.5:7b-instruct language model accessed
 1516 via the Ollama API. Following the methodology, we employ nine distinct prompt templates:

1517

```
1518 1 candidates = [
  2     """Let's think step by step."""
```

```

1512 3      """First,"""
1513 4      """The answer is after the proof."""
1514 5      """Before we dive into the answer,"""
1515 6      """Let's solve this problem by splitting it into steps."""
1516 7      """Let's think about this logically."""
1517 8      """It's a beautiful day."""
1518 9      """Don't think. Just feel."""
1519 10     """By the fact that the earth is round,"""
1520 11     ]

```

Listing 9: IAP Prompt Candidates

1522 For each prompt-question pair, we compute a composite saliency score:

$$S = \lambda_1 I_{qp} + \lambda_2 I_{qr} + \lambda_3 I_{pr} \quad (10)$$

1525 where:

- 1527 • I_{qp} : Question-to-prompt information flow
- 1528 • I_{qr} : Question-to-rationale information flow
- 1529 • I_{pr} : Prompt-to-rationale information flow
- 1530 • $\lambda_1 = 0.4, \lambda_2 = 0.4, \lambda_3 = 0.2$: Weighting parameters (summing to 1)

1533 A.6.3 MAJORITY VOTE (IAP-MV)

- 1534 1. Generate responses using all nine prompts
- 1535 2. Select top-3 responses by composite saliency score S
- 1536 3. Apply majority voting on the extracted answers
- 1537 4. Return the most frequent answer among top responses

1540 A.6.4 SEQUENTIAL SUBSTITUTION (IAP-ss)

- 1541 1. Iterate through prompts in predefined order
- 1542 2. For each prompt:
 - 1543 (a) Generate response and compute S
 - 1544 (b) If $S \geq \theta$ (threshold = 5.5×10^{-6}), return answer
- 1545 3. If no prompt meets threshold, return last generated answer

1548 A.7 USE OF LARGE LANGUAGE MODELS

1550 In accordance with the ICLR 2026 policy on the disclosure of Large Language Models (LLMs),
1551 we detail our use of LLMs in the preparation of this paper. LLMs were employed in a supportive
1552 capacity only, and their contributions were limited to non-core aspects of the work. All scientific
1553 content, including the conceptualization of the Holon-of-Thought (**HoT**) framework, methodology
1554 design, experimental setup, data analysis, and conclusions, was entirely developed by the human
1555 authors.

1556 We used LLMs (specifically GPT-4) to aid in polishing the writing of the manuscript. LLMs were
1557 prompted to suggest improvements to sentence structure, clarity, and grammatical accuracy in drafts
1558 of sections such as the abstract, introduction, related work, and methodology. For example, we
1559 provided raw paragraphs and asked the model to rephrase for conciseness while preserving the
1560 original meaning.

1561 LLMs were also used for retrieval and discovery purposes to identify potential related methods and
1562 literature. LLMs assisted in generating lists of potential citations by summarizing search queries
1563 related to “robustness in LLM reasoning” or “prompt-based decomposition techniques.” This accel-
1564 erated the discovery process, allowing us to quickly identify key works.