BEYOND SEMANTIC SIMILARITY: ABSTRACTION-BASED DEMONSTRATION RETRIEVAL FOR MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Selecting useful demonstrations is crucial for in-context mathematical reasoning with large language models (LLMs). However, using the original problem for retrieval often focuses too much on surface wording while ignoring the underlying mathematical relations needed for reasoning. To address this, we propose MetaPR, a retrieval-aligned meta-problem framework that learns to generate structured abstractions for retrieving demonstrations with reusable solution structures. For each problem, MetaPR generates a structured abstraction that combines a compact meta-problem with a concept list, preserving key constraints and mathematical relations while filtering out irrelevant details. The structured abstraction is combined with the original problem to form a multi-view retrieval representation. To ensure the generated abstractions genuinely help retrieval rather than merely being well-formed, the generator is initialized via supervised fine-tuning (SFT) and then optimized with Group Relative Policy Optimization (GRPO). During GRPO, we assign a Correctness-transition reward to each sampled abstraction by comparing answer correctness with and without the demonstrations it retrieves. Experiments across two NuminaMath-derived datasets and three downstream solvers demonstrate that MetaPR outperforms lexical, dense, skill-based, and reasoning-graph retrieval baselines in answer accuracy.

1 INTRODUCTION

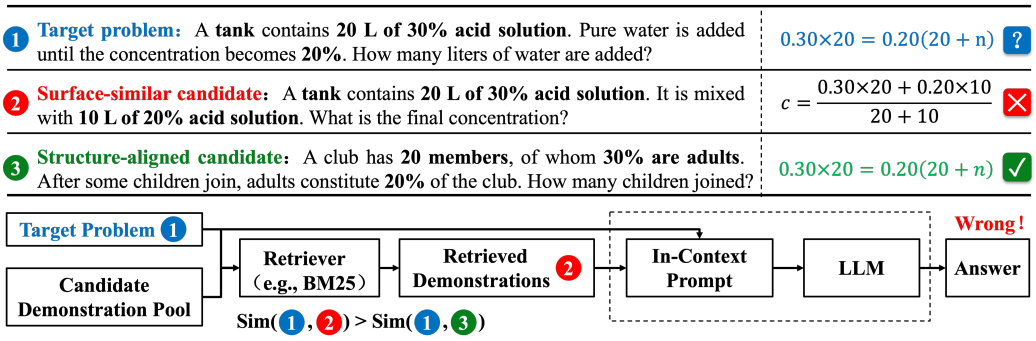


Figure 1: Motivation for Abstraction-Based Demonstration Retrieval. Given the target problem 1, a raw-text retriever may favor candidate 2, which shares the same tank-and-solution setting but follows a different mathematical relation. Candidate 3 has a different setting but shares the target's solution structure: $0.30 \times 20 = 0.20(20 + n)$. This contrast motivates retrieving demonstrations based on abstract, reusable solution structures.

Large Language Models (LLMs) have recently achieved remarkable progress in complex multi-step reasoning, particularly within mathematical domains. This progress is largely driven by In-Context Learning (ICL) (Wei et al., 2022; Wang et al., 2023a; Gao et al., 2023a). However, the performance of ICL is fragile and highly sensitive to the choice of few-shot examples. Poorly chosen or irrelevant examples can disrupt the reasoning of models and cause a large drop in accuracy.

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

To address this, retrieval-based demonstration selection has become a promising approach. Instead of using static, hand-crafted prompts, it retrieves the most relevant examples from a candidate pool for each input, which has been shown to improve results (Liu et al., 2022; Luo et al., 2023). However, most existing methods use the raw problem text as the retrieval query, so examples are retrieved based on semantic similarity. This is especially problematic for math problems, where surface similarity does not capture the similarity of the underlying reasoning. As illustrated in Figure 1, a retriever based on raw text may prefer a problem that looks similar but requires a different mathematical relation. Conversely, a problem expressed through a different narrative can share the same mathematical structure as the target and thus provide a more transferable reasoning pattern. This mismatch suggests that demonstrations should be retrieved based on their underlying solution structure rather than surface semantic similarity.

Following this insight, we introduce MetaPR, as shown in Figure 2, which treats structured abstraction as a learnable interface for mathematical demonstration retrieval. For each problem, the abstraction generator produces one structured representation with two complementary parts. The meta-problem preserves the target quantity, necessary constraints, and mathematical relations, while the concept list summarizes the relevant concepts and solution strategies. MetaPR combines these two parts with the original problem to form a multi-view retrieval key. The generated abstractions are used for retrieval of candidate demonstrations from the pool. The frozen solver will receive the original problem and the retrieved demonstrations to obtain the final answer.

Furthermore, a well-formed abstraction is not necessarily useful for retrieval. Faithful abstractions of the same problem may retrieve different demonstrations, leading to different reasoning results. We therefore train the abstraction generator using a two-stage retrieval-aligned procedure. First, supervised fine-tuning (SFT) teaches the generator to produce faithful abstractions under a fixed schema. Then, group relative policy optimization (GRPO) samples alternative abstractions and evaluates them through the demonstrations they retrieve. The reward aggregates whether these demonstrations improve or harm final-answer correctness across a frozen reward-solver ensemble. In this way, abstraction generation is optimized for retrieval utility rather than linguistic plausibility alone.

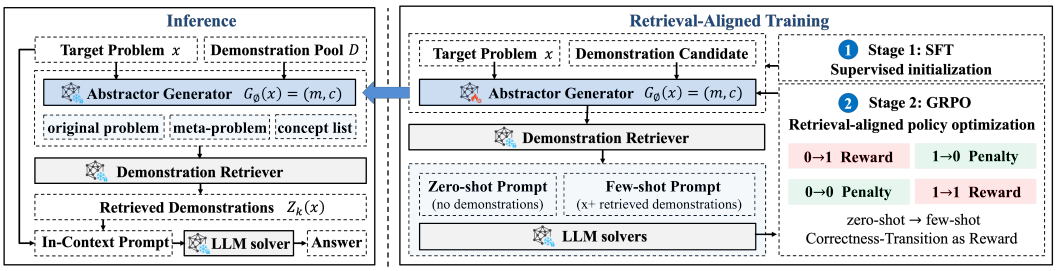


Figure 2: Illustration of MetaPR. The original problem, the generated meta-problem, and the generated concept list form a multi-view representation for retrieving demonstrations. After supervised initialization, the abstraction generator is optimized with GRPO using a correctness-transition reward. Each sampled abstraction is rewarded according to the retrieval-induced change in answer correctness relative to zero-shot solving.

Our contributions are summarized as follows:

1. **Solution-oriented demonstration retrieval.** We introduce MetaPR, a framework that enhances demonstration retrieval using structured abstractions. By combining the original problem with a meta-problem and a concept list, this multi-view representation effectively captures underlying mathematical structures and constraints beyond surface text similarity.
2. **Retrieval-utility alignment.** In MetaPR, we formulate the abstraction generation as a retrieval-aligned learning problem. The generator is first initialized via supervised fine-tuning and then optimized with GRPO, using the solver ensemble's aggregate final-answer correctness transition as direct feedback to maximize downstream retrieval utility.
3. **Systematic empirical validation.** MetaPR achieves advanced mathematical reasoning performance over lexical, dense, skill-based, and reasoning-graph retrieval baselines, across two NuminaMath-derived datasets, two retrieval-pool settings, and three frozen solvers.

2 Related Work

In-Context Learning. LLMs have demonstrated strong in-context learning ability, where models solve new tasks by conditioning on a few demonstrations without parameter updates (Brown et al., 2020; Chowdhery et al., 2022; OpenAI, 2023). For reasoning-intensive tasks, chain-of-thought prompting further improves ICL by providing intermediate reasoning steps in the context (Wei et al., 2022). Other methods use self-consistency decoding (Wang et al., 2023a), problem decomposition (Zhou et al., 2023), search over reasoning steps (Yao et al., 2023), and program-aided reasoning (Gao et al., 2023a). Studies of calibration and demonstration ordering also show that prompt design affects performance (Zhao et al., 2021; Lu et al., 2022). Together, these works show that few-shot reasoning benefits from structured reasoning steps and careful prompt design.

Demonstration Selection and Retrieval. Demonstration selection uses similarity, heuristics, or learned scores (Luo et al., 2024). BM25 uses lexical matching (Robertson & Zaragoza, 2009), KATE retrieves examples with similar embeddings (Liu et al., 2022), and dense encoders such as BGE-M3 provide embeddings for retrieval (Chen et al., 2024). Auto-CoT, Complexity-CoT, and Skill-KNN prioritize cluster-level diversity, rationale complexity, and LLM-generated skill representations, respectively (Zhang et al., 2023; Fu et al., 2023; An et al., 2023). Learned retrievers derive supervision from downstream LLM feedback: EPR and LLM-R optimize dense retrieval models, CEIL captures set-level interactions among exemplars, and UDR and MDR target cross-task transfer and inference-model-specific utility, respectively (Rubin et al., 2022; Wang et al., 2024a; Ye et al., 2023; Li et al., 2023; Wang et al., 2024b). Reasoning-aware approaches enrich retrieval representations with generated knowledge, computational graphs, and reasoning-derived supervision (Xiong et al., 2024; Yang et al., 2024a; Das et al., 2025). In particular, GraphIC uses graph representations for example retrieval (Fu et al., 2026), and RGER combines semantic and reasoning-graph similarity (Lin et al., 2025). HyDE, Query2doc, Rewrite-Retrieve-Read, and Step-Back Prompting instead reformulate queries through hypothetical-document generation, query expansion, learned rewriting, or abstraction (Gao et al., 2023b; Wang et al., 2023b; Ma et al., 2023; Zheng et al., 2024). Together, these approaches identify solution-relevant structure and downstream usefulness as complementary considerations in demonstration retrieval.

Reinforcement Learning for Retrieval and Reasoning. DeepSeekMath introduced GRPO, and DeepSeek-R1-Zero showed that reinforcement learning (RL) can improve reasoning without initial supervised fine-tuning (Shao et al., 2024; Guo et al., 2025). For retrieval, DeepRetrieval and MaFeRw use RL to generate or rewrite queries, while Search-R1 and R1-Searcher learn to use search during reasoning (Jiang et al., 2025; Wang et al., 2025a; Jin et al., 2025; Song et al., 2025). For demonstrations, RL-ICL, RDES, and reinforced compositional retrieval optimize selection policies, whereas GRAD learns to generate input-specific demonstrations using downstream feedback (Long et al., 2024; Wang et al., 2025b; Long et al., 2025; Gabouj et al., 2025). RLAD and AbstRaL further apply RL to abstraction learning: RLAD learns reasoning strategies that guide a solver, while AbstRaL learns mathematical abstractions for symbolic reasoning (Qu et al., 2026; Gao et al., 2026). We instead use RL to learn mathematical abstractions for demonstration retrieval, with rewards based on the final-answer correctness achieved using the retrieved examples.

3 Method

3.1 Problem Formulation

Given a target mathematical problem x, we consider a demonstration pool $\mathcal{D} = \{d_i\}_{i=1}^{N}$, where each demonstration $d_i = (x_i, r_i, y_i)$ consists of a problem statement x_i, a reasoning trace r_i, and the final answer y_i. A demonstration retriever selects a subset of k examples, denoted by $\mathcal{Z}_k(x) \subset \mathcal{D}$, according to their relevance to the target problem. The retrieved demonstrations are then combined with x to construct a few-shot prompt for a frozen LLM solver M, which produces the predicted answer $\hat{y} = M(\mathrm{Prompt}(x, \mathcal{Z}_k(x)))$. Our objective is to retrieve demonstrations that provide reusable solution guidance, thereby maximizing the probability that $\hat{y}$ matches the ground-truth answer y. The key challenge is that relevance for mathematical reasoning is determined not only by semantic similarity between problem statements, but, more importantly, by whether the target problem and the retrieved demonstrations share compatible mathematical relations and solution structures.

3.2 Reasoning-Oriented Abstractions

Retrieving demonstrations by text similarity often captures surface wording rather than core mathematical structure. To address this limitation, the proposed abstraction generator G_ϕ maps each problem x to a structured abstraction a comprising a meta-problem m and a concept list c:

$$a = G_\phi(x) = (m, c). \tag{1}$$

Meta problem is an abstract reformulation of the original problem that removes incidental entities and narrative details while preserving the specific quantity, necessary conditions, and solution-relevant mathematical relations. For example, "Five kilograms of apples cost eight dollars per kilogram. What is the total cost?" can be reformulated as "Given a unit price and a quantity, compute the total cost." By exposing the underlying relational structure, the meta-problem enables retrieval across problems with different surface contexts but similar solution patterns.

Concept list is a concise set of mathematical concepts, operations, and strategies that are potentially involved in solving the problem. For the example above, the concept list may include "multiplication," "unit rate," and "total cost." While the meta-problem captures the overall relational structure, the concept list provides explicit concept-level cues that help distinguish problems requiring different mathematical knowledge or operations.

3.3 Abstraction-based Demonstration Retrieval

To find better demonstrations, retrieval representations should capture both mathematical structure and problem-specific details. While the meta-problem and concept list reflect core concepts and relations, the original problem offers concrete context. We therefore build a multi-view representation that combines the original problem, the meta-problem, and the concept list.

Let $E(\cdot)$ denote a frozen text encoder and $e(t) = E(t)/\|E(t)\|_2$ denote its normalized embedding. For a target problem x with abstraction $a = (m, c)$ and a candidate demonstration $d_i = (x_i, r_i, y_i)$ with abstraction $a_i = (m_i, c_i)$, we compute a weighted sum of same-view similarities:

$$s(x, a; d_i, a_i) = \alpha_x \langle e(x), e(x_i)\rangle + \alpha_m \langle e(m), e(m_i)\rangle + \alpha_c \langle e(\mathrm{Serialize}(c)), e(\mathrm{Serialize}(c_i))\rangle, \tag{2}$$

where $\mathrm{Serialize}(c)$ joins the concept strings into text, and α_x, α_m, and α_c control the contributions of the original problem, meta-problem, and concept list, respectively.

For each candidate demonstration $d_i = (x_i, r_i, y_i)$, its abstraction $a_i = (m_i, c_i)$ and the three view embeddings are precomputed and stored in the retrieval index. At inference time, we encode the target views with the same encoder and select the k highest-scoring demonstrations:

$$\begin{aligned} \mathcal{I}_k(x, a) &= \underset{i \in \{1,\ldots,N\}}{\arg\operatorname{topk}}\ s(x, a; d_i, a_i), \\ \mathcal{Z}_k(x, a) &= \{d_i \mid i \in \mathcal{I}_k(x, a)\}, \end{aligned} \tag{3}$$

where $\mathcal{I}_k(x, a)$ denotes the selected candidate indices of the demonstration pool. Following Section 3.1, $\mathcal{Z}_k(x, a)$ is combined with x to construct a few-shot prompt for the frozen solver M.

3.4 Retrieval-Aligned Abstraction Learning

The retrieval framework above relies on the abstraction generator G_ϕ to produce informative meta-problems and concept lists. However, a well-formed abstraction is not necessarily effective for retrieval: different valid abstractions of the same problem may retrieve different demonstrations and lead to substantially different reasoning outcomes. Therefore, G_ϕ should be explicitly aligned with downstream solver performance. To this end, G_ϕ is trained in two stages, *i.e.* first to produce valid abstractions, and then to optimize them to maximize the solver's reasoning performance.

Stage 1: Supervised initialization. The first stage aims to initialize G_ϕ with the ability to generate well-formed meta-problems and concept lists. Let $\mathcal{Q}_{\mathrm{train}} = \{x_j\}_{j=1}^{N_{\mathrm{train}}}$ denote the training set for G_ϕ. To construct the supervision data, we design a structured prompt that instructs a strong teacher model $\mathcal{T}$ to produce the abstractions under a predefined output schema. Specifically, for each $x_j \in \mathcal{Q}_{\mathrm{train}}$, the teacher model $\mathcal{T}$ produces a reference abstraction,

$$a_j^\star = \mathcal{T}(x_j) = (m_j^\star, c_j^\star), \tag{4}$$

where $a_j^\star$ is the corresponding teacher-generated abstraction consisting of the meta-problem $m_j^\star$ and the concept list $c_j^\star$. The pairs $(x_j, a_j^\star)$ are used to initialize G_ϕ through SFT

$$\mathcal{L}_{\mathrm{SFT}}(\phi) = -\sum_j \sum_{t=1}^{|a_j^\star|} \log \pi_\phi \left(a_{j,t}^\star \mid x_j, a_{j,<t}^\star\right), \tag{5}$$

where π_ϕ denotes the autoregressive policy induced by G_ϕ, $|a_j^\star|$ is the token length of the serialized teacher abstraction, $a_{j,t}^\star$ is its t-th token, and $a_{j,<t}^\star$ is the preceding token sequence.

Stage 2: Retrieval-aligned policy optimization. Although SFT enables G_ϕ to generate well-formed abstractions, it does not optimize abstractions for downstream retrieval effectiveness. We therefore further optimize the generator using GRPO, with the final reasoning outcome as feedback.

Candidate generation. For each training problem x_j, from the current policy denoted $\pi_{\phi_{\mathrm{old}}}$, we sample a group of G candidate abstractions

$$a_j^{(g)} \sim \pi_{\phi_{\mathrm{old}}}(\cdot \mid x_j), \quad g \in \{1, \ldots, G\}, \tag{6}$$

where each candidate $a_j^{(g)} = (m_j^{(g)}, c_j^{(g)})$ contains a meta-problem $m_j^{(g)}$ and a concept list $c_j^{(g)}$. Sampling multiple candidates promotes exploration over diverse abstractions of the same problem.

Downstream utility evaluation. Each candidate abstraction $a_j^{(g)}$ is combined with the original problem x_j to score the three retrieval views. We then retrieve $\mathcal{Z}_{k,j}^{(g)} = \mathcal{Z}_k(x_j, a_j^{(g)})$ according to Eq. (3). Let $\mathcal{M}_{\mathrm{RL}}$ denote the frozen reward-solver ensemble used throughout GRPO, comprising Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, and GLM-4-9B-0414. For each $M \in \mathcal{M}_{\mathrm{RL}}$, the zero-shot prediction $\hat{y}_{j,M}^{(0)}$ and retrieval-augmented prediction $\hat{y}_{j,M}^{(g)}$ are given by

$$\hat{y}_{j,M}^{(0)} = M(\mathrm{Prompt}(x_j, \varnothing)), \quad \hat{y}_{j,M}^{(g)} = M(\mathrm{Prompt}(x_j, \mathcal{Z}_{k,j}^{(g)})), \tag{7}$$

where $\mathcal{Z}_{k,j}^{(g)}$ denotes the demonstrations retrieved using the g-th candidate abstraction. Given the ground-truth answer y_j, we quantify retrieval utility via a correctness-transition reward:

$$r_{\mathrm{ct}}(\hat{y}_{j,M}^{(0)}, \hat{y}_{j,M}^{(g)}, y_j) = \begin{cases} r_{01}, & \hat{y}_{j,M}^{(0)} \neq y_j \text{ and } \hat{y}_{j,M}^{(g)} = y_j, \\ r_{11}, & \hat{y}_{j,M}^{(0)} = y_j \text{ and } \hat{y}_{j,M}^{(g)} = y_j, \\ r_{10}, & \hat{y}_{j,M}^{(0)} = y_j \text{ and } \hat{y}_{j,M}^{(g)} \neq y_j, \\ r_{00}, & \hat{y}_{j,M}^{(0)} \neq y_j \text{ and } \hat{y}_{j,M}^{(g)} \neq y_j. \end{cases} \tag{8}$$

The correctness-transition reward encourages abstractions that turn an incorrect zero-shot prediction into a correct retrieval-augmented prediction, while penalizing harmful retrieval that changes a correct prediction into an incorrect one. We sum this signal across the three reward solvers. We further use a binary format reward r_{fmt} for schema compliance and a soft length penalty p_{len} to discourage verbose outputs. The overall reward for the g-th candidate is

$$r_j^{(g)} = \sum_{M \in \mathcal{M}_{\mathrm{RL}}} r_{\mathrm{ct}}\left(\hat{y}_{j,M}^{(0)}, \hat{y}_{j,M}^{(g)}, y_j\right) + \lambda_{\mathrm{fmt}} r_{\mathrm{fmt},j}^{(g)} - \lambda_{\mathrm{len}} p_{\mathrm{len},j}^{(g)}. \tag{9}$$

where the coefficients λ_{fmt} and λ_{len} control the format bonus and length penalty. We use $(r_{01}, r_{11}, r_{10}, r_{00}) = (1, 0.05, -1, -0.1)$ and $\lambda_{\mathrm{fmt}} = \lambda_{\mathrm{len}} = 0.5$. Appendix C specifies the schema and soft-penalty terms.

Policy optimization. Candidate rewards are normalized within each group to compute advantages:

$$A_j^{(g)} = \frac{r_j^{(g)} - \bar{r}_j}{\sqrt{\frac{1}{G}\sum_{u=1}^{G}(r_j^{(u)} - \bar{r}_j)^2 + \epsilon}}, \qquad \bar{r}_j = \frac{1}{G}\sum_{u=1}^{G} r_j^{(u)}. \tag{10}$$

where $\bar{r}_j$ is the mean reward of the candidate group and ϵ prevents division by zero. Following (Shao et al., 2024), we optimize π_ϕ using the clipped surrogate loss with a KL penalty against the frozen π_{ref}, steering G_ϕ to produce abstractions that yield effective demonstrations for downstream solving.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and evaluation settings. We construct two datasets from the *aops_forum* and *olympiads* subsets of NuminaMath-1.5 (Project Numina, 2024). First, we remove diagram-dependent problems, problems without reliable final answers, multi-target questions, and proof-only tasks, and normalize the remaining samples into problem–reasoning–answer triples. Next, we use clusters of problem texts to guide a 70:30 partition into an initial demonstration pool and a query pool, and immediately divide the query pool into fixed training and test sets at an approximate 80:20 ratio. To model variation in how structurally related problems are expressed, we then rewrite selected problems from the already partitioned query pool while preserving their solution structure and add the accepted rewritten triples to the shared demonstration pool. Thus, augmentation does not alter train/test membership, and both query sets use the same augmented demonstration pool. For both source datasets (*aops_forum* and *olympiads*), the initial split contains approximately 3,200 training queries, 800 test queries, and 9,300 demonstrations. Appendix A details the rewriting strategies.

For evaluation, we divide the test queries into two disjoint subsets. B (matched) contains queries for which at least one structure-preserving rewrite was deliberately added to the demonstration pool, whereas A (uncontrolled) contains the remaining queries. Naturally occurring structural matches may still exist in A. ALL denotes $A \cup B$. These labels describe query subsets sharing one pool, rather than separate retrieval databases.

Evaluation metrics. We report final-answer accuracy (Acc.) and Hit@3 for the deliberately constructed rewrites. Let $\hat{y}$ be the model prediction and y be the gold answer. Accuracy is

$$\text{Acc.} = \frac{1}{|\mathcal{T}|} \sum_{x \in \mathcal{T}} \mathbb{I}[\mathcal{V}(x, y, \hat{y}) = \text{EQUIVALENT}]. \tag{11}$$

The judge $\mathcal{V}$ compares the reference and model output under a binary EQUIVALENT/NOT-EQUIVALENT protocol. Missing answers and outputs without a parseable verdict are counted as NOT-EQUIVALENT. Hit@3 measures the fraction of queries in B whose top-three results contain at least one rewrite constructed from that query; its formula and results are given in Appendix B.1.

Models and hyperparameters. We evaluate with multiple solvers, including Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct. The generator backbone is Qwen2.5-7B-Instruct (Yang et al., 2024b), initialized through SFT on approximately 3,200 teacher-generated examples from DeepSeek-V3.2 (DeepSeek-AI, 2025). We use BGE-M3 (Chen et al., 2024) as the frozen dense encoder, set $\alpha_x = \alpha_m = \alpha_c = 1/3$, and retrieve three demonstrations in the main experiments. For BM25, we lowercase the English problem text and apply word-level tokenization. To ensure a fair comparison, all compared methods use the same demonstration pool, prompt template, demonstration budget, solver decoding configuration, and answer-matching script. Unless otherwise specified, solver inference uses greedy decoding to reduce sampling variance.

Baselines. We compare MetaPR with Zero-shot, Random, original-query retrievers, and reasoning-aware methods. **Zero-shot** uses no demonstrations, while **Random** uniformly samples k demonstrations from the pool. **BM25** and **Dense** retrieve demonstrations using lexical matching and dense embeddings of the original problem text, respectively. Among reasoning-aware retrieval methods, **Auto-CoT** (Zhang et al., 2023) selects cluster representatives, **Complexity-CoT** (Fu et al., 2023) prioritizes complex reasoning chains, and **Skill-KNN** (An et al., 2023) retrieves with LLM-generated skill descriptions. **GraphIC** (Fu et al., 2026) represents reasoning structure with graphs, whereas **RGER** (Lin et al., 2025) combines semantic similarity with reasoning-graph similarity.

4.2 MAIN RESULTS

Table 1 reports final-answer accuracy for all comparison methods. MetaPR achieves the best ALL accuracy for every solver on both datasets. With Qwen2.5-7B-Instruct, it reaches 30.88% on *aops_forum*, improving over the strongest baseline, GraphIC, by 1.88 points; on *olympiads*, it reaches 62.29%, 0.37 points above GraphIC. With Llama3.1-8B-Instruct, MetaPR improves over GraphIC by 1.62 points on *aops_forum* and 2.70 points on *olympiads*. With Llama3.2-3B-Instruct, the corresponding gains over the strongest baselines are 1.87 and 1.96 points.

In Table 1, we also observe substantial variation across retrieval strategies. Random retrieval can be harmful, BM25 and dense retrieval can exchange rank across datasets, and GraphIC and RGER

Table 1: Main results on two NuminaMath-derived datasets. A and B denote matched and uncontrolled settings, respectively; ALL denotes their union. Best results per solver are bold.

Solver	Method	*aops_forum* Acc.			*olympiads* Acc.		
		A	B	ALL	A	B	ALL
Llama-3.1-8B Instruct	Zero-shot	11.79%	5.83%	10.00%	42.11%	45.90%	43.24%
	Random	9.11%	4.58%	7.75%	40.00%	44.26%	41.28%
	Auto-CoT (Zhang et al., 2023)	19.64%	21.25%	20.13%	45.61%	47.13%	46.07%
	Complexity-CoT (Fu et al., 2023)	18.75%	19.17%	18.88%	46.14%	51.64%	47.79%
	BM25 (Robertson & Zaragoza, 2009)	20.54%	39.17%	26.13%	47.02%	53.28%	48.89%
	Dense (Chen et al., 2024)	19.29%	37.50%	24.75%	46.32%	52.87%	48.28%
	Skill-KNN (An et al., 2023)	24.46%	40.83%	29.38%	50.00%	58.61%	52.58%
	GraphIC (Fu et al., 2026)	26.43%	47.08%	32.63%	52.63%	60.66%	55.04%
	RGER (Lin et al., 2025)	24.64%	**48.33%**	31.75%	51.58%	62.30%	54.79%
	MetaPR	**28.93%**	46.67%	**34.25%**	**54.39%**	**65.57%**	**57.74%**
Qwen2.5-7B Instruct	Zero-shot	24.46%	20.83%	23.38%	55.26%	59.84%	56.63%
	Random	24.64%	21.25%	23.63%	54.39%	58.61%	55.65%
	Auto-CoT (Zhang et al., 2023)	24.82%	22.50%	24.13%	56.84%	61.07%	58.11%
	Complexity-CoT (Fu et al., 2023)	25.18%	27.50%	25.88%	57.89%	56.97%	57.62%
	BM25 (Robertson & Zaragoza, 2009)	23.57%	23.33%	23.50%	58.77%	61.07%	59.46%
	Dense (Chen et al., 2024)	21.96%	23.33%	22.38%	57.19%	63.11%	58.97%
	Skill-KNN (An et al., 2023)	26.61%	28.75%	27.25%	59.82%	61.07%	60.20%
	GraphIC (Fu et al., 2026)	28.21%	30.83%	29.00%	60.70%	64.75%	61.92%
	RGER (Lin et al., 2025)	27.50%	30.00%	28.25%	59.47%	**65.98%**	61.43%
	MetaPR	**29.64%**	**33.75%**	**30.88%**	**61.05%**	65.16%	**62.29%**
Llama-3.2-3B Instruct	Zero-shot	11.61%	8.33%	10.63%	32.46%	33.20%	32.68%
	Random	4.29%	4.58%	4.38%	29.47%	30.74%	29.85%
	Auto-CoT (Zhang et al., 2023)	12.50%	17.50%	14.00%	34.39%	34.43%	34.40%
	Complexity-CoT (Fu et al., 2023)	12.14%	15.83%	13.25%	35.79%	35.25%	35.63%
	BM25 (Robertson & Zaragoza, 2009)	15.18%	27.08%	18.75%	37.72%	38.52%	37.96%
	Dense (Chen et al., 2024)	13.04%	30.83%	18.38%	37.02%	39.34%	37.71%
	Skill-KNN (An et al., 2023)	18.04%	33.33%	22.63%	40.70%	41.39%	40.91%
	GraphIC (Fu et al., 2026)	18.75%	37.92%	24.50%	41.58%	46.31%	43.00%
	RGER (Lin et al., 2025)	19.46%	39.17%	25.38%	42.46%	43.03%	42.63%
	MetaPR	**20.89%**	**42.08%**	**27.25%**	**43.51%**	**48.36%**	**44.96%**

Table 2: Ablation of abstraction-generator training stages. A and B denote matched and uncontrolled settings, respectively; ALL denotes their union. Best results within each solver block are bold.

Solver	Training Stage	*aops_forum* Acc.			*olympiads* Acc.		
		A	B	ALL	A	B	ALL
Llama-3.1-8B Instruct	Base	21.96%	39.58%	27.25%	48.07%	57.79%	50.98%
	SFT	**29.11%**	38.33%	31.88%	51.58%	61.89%	54.67%
	GRPO	28.93%	**46.67%**	**34.25%**	**54.39%**	**65.57%**	**57.74%**
Qwen2.5-7B Instruct	Base	24.11%	22.08%	23.50%	56.14%	62.30%	57.99%
	SFT	25.36%	**34.17%**	28.00%	60.00%	62.70%	60.81%
	GRPO	**29.64%**	33.75%	**30.88%**	**61.05%**	**65.16%**	**62.29%**
Llama-3.2-3B Instruct	Base	15.18%	30.42%	19.75%	36.84%	41.39%	38.21%
	SFT	**22.14%**	30.42%	24.63%	40.00%	**49.18%**	42.75%
	GRPO	20.89%	**42.08%**	**27.25%**	**43.51%**	48.36%	**44.96%**

are strongest on different solver–split combinations. This supports our motivation: adding demonstrations is insufficient unless the retrieved examples match the target problem's solution logic. By using generated abstractions and retrieval-aligned training, MetaPR provides the most consistent ALL accuracy across solvers and datasets.

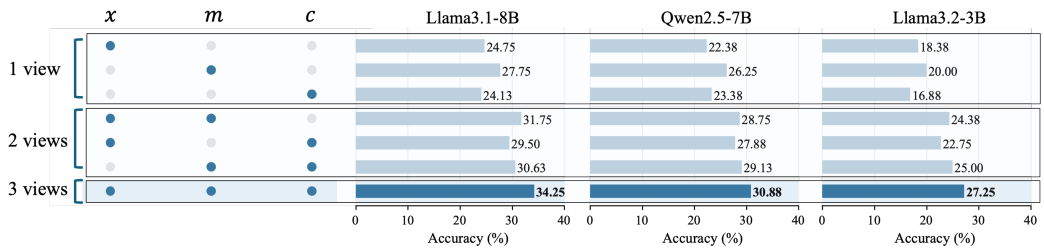


Figure 3: Retrieval-view ablation on *aops_forum* (ALL). x, m and c represent the raw problem, meta-problem and concept list, respectively. The full three-view combination (x+m+c) consistently outperforms all one-view and two-view alternatives.

Table 3: Examples from the top-three retrieved demonstrations. Each pair connects different surface contexts through a shared solution structure. Rank gives the position in the retrieved list. Problem statements are condensed, and **bold** marks related constraints, targets, and concepts.

(a) Reciprocal-square Diophantine equation

Query	**Retrieved demonstration** Rank 1
A chemistry experiment mixes two positive-integer volumes a, b and requires $\mathbf{1/p = 1/a^2 + 1/b^2}$, where p is prime. Find all **triplets** (a, b, p).	An image-compression scheme uses positive-integer square-tile sides a, b with data density $\mathbf{1/p = 1/a^2 + 1/b^2}$, where p is prime. Determine all **triplets** (a, b, p).
meta_problem Given positive integers a, b and a prime p, find all triplets satisfying the **reciprocal-square equation** $1/p = 1/a^2 + 1/b^2$.	**meta_problem** Given two positive integers and a prime parameter, determine all triplets satisfying a **reciprocal-square Diophantine equation**.
concept_list **Diophantine equations**; prime numbers; **reciprocal relationships**; number theory	**concept_list** **Diophantine equations**; reciprocal sums; prime numbers; **integer solutions**

(b) Factorial trailing zeros and an arithmetic-progression product

Query	**Retrieved demonstration** Rank 3
Find the integer $n \geq 48$ such that $n!$, when written in decimal notation, **ends with exactly** $n - 48$ **consecutive zeros**.	Let $k = 2 \cdot 5 \cdot 8 \cdot 11 \cdots 2006$, the product of every third positive integer from 2 to 2006. Find the number of **trailing zeros** of k.
meta_problem Given an integer $n \geq 48$, find n such that the **number of trailing zeros in** $n!$ equals $n - 48$.	**meta_problem** Given a product of terms in an **arithmetic progression**, count its **number of trailing zeros**.
concept_list **trailing zeros**; divisibility by 10; **counting factors of** 2 **and** 5	**concept_list** arithmetic progression; **counting factors of** 2 **and** 5; prime factorization; **trailing zeros**

(c) Cubic-polynomial objectives and Vieta's formulas

Query	**Retrieved demonstration** Rank 2
In the cubic $T^3 + aT^2 + bT + c$, all roots are positive integers greater than 2 and $a + b + c + 1 = -2009$. Determine a.	Let f be a monic cubic whose coefficient sum is 5 and whose root sum is 1. Find the absolute value of the **sum of the cubes of its roots**.
meta_problem Given a cubic polynomial with positive integer roots greater than 2 and a coefficient-sum constraint, find the **quadratic coefficient**.	**meta_problem** Given a monic cubic with specified coefficient and root sums, find the absolute value of a **symmetric power sum of its roots**.
concept_list **cubic polynomial**; **Vieta's formulas**; positive integer roots; symmetric polynomials	**concept_list** **monic cubic**; **Vieta's formulas**; symmetric root identities; sums of powers

4.3 Ablation Study

We examine how generator training and the choice of retrieval views affect reasoning performance.

Generator training. Table 2 compares the Base, SFT, and GRPO stages of the Qwen2.5-7B-Instruct generator while keeping the retrieval and solver configurations fixed. The results show a consistent improvement in ALL accuracy from the base generator to SFT and then to the GRPO-optimized generator across all solvers and datasets, although individual A and B splits exhibit small trade-offs. For the Qwen2.5-7B-Instruct solver, SFT improves *aops_forum* ALL accuracy by 4.50 points over the base generator, and GRPO adds another 2.88 points. On *olympiads* ALL, the corresponding gains are 2.82 and 1.48 points. Similar ALL-level trends for both Llama solvers indicate that schema-consistent abstraction is helpful, while downstream correctness feedback further aligns the generator with examples that improve solving.

Retrieval views. Figure 3 compares the 7 nonempty combinations of the raw problem (x), meta-problem (m) and concept list (c): 3 one-view variants (x, m, c), 3 two-view variants ($x+m$, $x+c$, $m+c$), and the full $x+m+c$ representation. Every two-view variant outperforms the best one-view for each solver. The best pair is $x+m$ for Llama3.1-8B-Instruct and $m+c$ for Qwen2.5-7B-Instruct and Llama3.2-3B-Instruct. Combining all three views improves over these pairs by 2.50, 1.75, and 2.25 points, respectively, supporting the complementary value of the three retrieval views.

4.4 Analysis

Why can original-query retrieval fail? The main results (Table 1) show that lexical and dense retrieval do not consistently improve over zero-shot solving. This suggests that raw problem statements are noisy retrieval queries for mathematical reasoning. Surface narratives and entities may dominate the representation even when they are irrelevant to the solution. The *meta-problem* representation reduces this noise by focusing on target unknowns, constraints, and mathematical relations.

Why does GRPO help beyond SFT? SFT teaches the generator to imitate teacher abstractions, but even the teacher-labeled abstractions are not necessarily optimal for the downstream solver. GRPO directly evaluates whether an abstraction retrieves demonstrations that improve final-answer accuracy (Table 2). The reward design also penalizes harmful retrieval cases where a correct zero-shot answer becomes incorrect after adding demonstrations.

Behavior across evaluation splits. MetaPR improves both A and B splits over original-query retrieval, indicating that it can exploit available solution-logic matches while remaining useful in less controlled pools. On the A split of *aops_forum*, Qwen2.5-7B-Instruct reaches 29.64% with MetaPR, compared with 28.21% for GraphIC, the strongest reasoning-aware baseline in this setting. This suggests that the learned abstraction is not merely memorizing constructed rewrites but provides a more robust query representation for demonstration retrieval. Appendix B.1 further evaluates whether each retriever recovers the deliberately constructed rewrites.

Qualitative analysis. Table 3 illustrates three forms of solution-oriented relevance. Pair (a) places the same reciprocal-square Diophantine equation in unrelated contexts. Pair (b) connects a factorial equation with an arithmetic-progression product through counting factors of 2 and 5. Pair (c) links different cubic-polynomial objectives through Vieta's formulas and symmetric root identities. Together, they show how the abstraction views expose reusable mathematical structure beyond narrative and lexical overlap.

5 Conclusion

We propose MetaPR, a retrieval-aligned meta-problem framework for few-shot mathematical reasoning. MetaPR represents each problem through the original statement, a *meta-problem*, and a concept list, thereby reducing surface-similarity bias and emphasizing reusable solution structure. It further trains the abstraction generator with SFT and GRPO, using downstream answer correctness as the optimization signal. Experiments on NuminaMath-derived datasets show that MetaPR improves final-answer accuracy over standard retrieval baselines, and ablations show that retrieval-aligned GRPO provides additional gains beyond supervised fine-tuning. These findings suggest that mathematical demonstration retrieval should be aligned with solution structure and actual solver feedback rather than surface text similarity alone.

AI USE STATEMENT

Generative AI tools were used to assist with literature search, language editing, manuscript organization, and formatting. The authors verified the cited sources and reviewed all AI-assisted text. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Appendix A documents data construction, Appendix C.1 reports the complete training and retrieval configuration, and Appendix C provides the abstraction, solver, and judge prompts.

REFERENCES

Shengnan An, Bo Zhou, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Weizhu Chen, and Jian-Guang Lou. 2023. Skill-based few-shot selection for in-context learning. In *Proceedings of EMNLP*, pages 13472–13492.

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

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. 2025. RaDeR: Reasoning-aware dense retrieval models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19982–20009.

DeepSeek-AI. 2025. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Jiale Fu, Yaqing Wang, Simeng Han, Jiaming Fan, and Xu Yang. 2026. GraphIC: A graph-based in-context example retrieval model for multi-step reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30753–30761.

Oussama Gabouj, Kamel Charaf, Ivan Zakazov, Nicolas Baldwin, and Robert West. 2025. GRAD: Generative retrieval-aligned demonstration sampler for efficient few-shot reasoning. In *Findings of EMNLP*, pages 19226–19244.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. PAL: Program-aided language models. In *Proceedings of ICML*, pages 10764–10799.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023b. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of ACL*, pages 1762–1777.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. DeepRetrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. In *The Second Conference on Language Modeling*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Proceedings of COLM*.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of ACL*, pages 4644–4668.

Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. 2024. MuggleMath: Assessing the impact of query and response augmentation on math reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10230–10258.

Yukang Lin, Bingchen Zhong, Shuoran Jiang, Joanna Siebert, and Qingcai Chen. 2025. Reasoning graph enhanced exemplars retrieval for in-context learning. In *Proceedings of COLING*, pages 9737–9759.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of DeeLIO*, pages 100–114.

Quanyu Long, Jianda Chen, Wenya Wang, and Sinno Jialin Pan. 2024. Large language models know what makes exemplary contexts. *arXiv preprint arXiv:2408.07505*.

Quanyu Long, Jianda Chen, Zhengyuan Liu, Nancy F. Chen, Wenya Wang, and Sinno Jialin Pan. 2025. Reinforcing compositional retrieval: Retrieving step-by-step for composing informative contexts. In *Findings of ACL*, pages 7633–7651.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of ACL*, pages 8086–8098.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y. Zhao. 2023. Dr.ICL: Demonstration-retrieved in-context learning. *arXiv preprint arXiv:2305.14128*.

Man Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. In *Proceedings of EMNLP*, pages 5303–5315.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Project Numina. 2024. NuminaMath-1.5 dataset card. Hugging Face repository.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of NAACL-HLT*, pages 2655–2671.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of EMNLP*, pages 9414–9423.

Liang Wang, Nan Yang, and Furu Wei. 2024a. Learning to retrieve in-context examples for large language models. In *Proceedings of EACL*, pages 1752–1767.

Huazheng Wang, Jinming Wu, Haifeng Sun, Zixuan Xia, Daixuan Cheng, Jingyu Wang, Qi Qi, and Jianxin Liao. 2024b. MDR: Model-specific demonstration retrieval at inference time for in-context learning. In *Proceedings of NAACL-HLT*, pages 4189–4204.

Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. 2025a. MaFeRw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. In *Proceedings of AAAI*, volume 39, pages 25434–25442.

Xubin Wang, Jianfei Wu, Yichen Yuan, Deyu Cai, Mingzhe Li, and Weijia Jia. 2025b. Demonstration selection for in-context learning via reinforcement learning. In *Proceedings of ICML*, volume 267, pages 64937–64954.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Jing Xiong, Zixuan Li, Chuanyang Zheng, Zhijiang Guo, Yichun Yin, Enze Xie, Zhicheng Yang, Qingxing Cao, Haiming Wang, Xiongwei Han, Jing Tang, Chengming Li, and Xiaodan Liang. 2024. DQ-LoRe: Dual queries with low-rank approximation re-ranking for in-context learning. In *Proceedings of ICLR*.

Xiaocong Yang, Jiacheng Lin, Ziqi Wang, and ChengXiang Zhai. 2024a. Learning by analogy: Enhancing few-shot prompting for math word problem solving with computational graph-based retrieval. *arXiv preprint arXiv:2411.16454*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 39818–39833.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of ICML*, pages 12697–12706.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *Proceedings of ICLR*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Table 4: Structure-preserving rewriting strategies used in dataset construction.

Strategy	Transformation	Consistency requirement
Contextual	Replace the application domain and narrative framing.	Preserve the mathematical core, constraints, answer type, and difficulty.
Synonym	Make only light lexical or sentence-pattern changes.	Preserve all symbols, quantities, constraints, and the answer.
Variable	Rename free variables throughout the problem.	Keep constants, functions, operators, units, and relations unchanged.
Numerical	Replace numerical constants with a valid alternative instance.	Preserve structural properties and difficulty; recompute the answer.

Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Setlur, Russ Salakhutdinov, Chelsea Finn, and Aviral Kumar. 2026. RLAD: Training LLMs to discover abstractions for solving reasoning problems. In *Proceedings of ICLR*.

Silin Gao, Antoine Bosselut, Samy Bengio, and Emmanuel Abbe. 2026. AbstRaL: Augmenting LLMs' reasoning by reinforcing abstract thinking. In *Proceedings of ICLR*.

A Dataset Construction Details

Partitioning and rewrite placement. The two NuminaMath-1.5 source subsets are processed independently. After filtering and normalization, each subset is partitioned into an initial demonstration pool and a query pool using a 70:30 split, and the query pool is immediately divided into fixed training and test queries using an approximate 80:20 split. This construction yields approximately 3,200 training queries, 800 test queries, and 9,300 demonstrations per source subset. Only after these memberships are fixed do we randomly select problems from the query pool for augmentation and add their accepted rewrites to the shared demonstration pool. A test query whose accepted rewrite is added to the pool is assigned to subset B; every other test query is assigned to subset A; and ALL is their union. The exact A/B/ALL test-query counts are 560/240/800 for *aops_forum* and 570/244/814 for *olympiads*.

Rewriting strategies. Table 4 summarizes the four transformations. All prompts require the rewritten problem to remain determinate and to preserve the mathematical structure and difficulty of the source problem. Contextual, synonym, and variable rewrites retain the original quantities and answer, whereas numerical substitution changes constants subject to the original mathematical constraints and therefore requires a newly computed answer.

Following MuggleMath's multi-path filtering practice (Li et al., 2024), we solve each rewrite multiple times and retain only those with consistent or mathematically equivalent final answers. Numerical rewrites receive newly generated solutions and answers. Section A.1 provides complete original–rewrite pairs.

Rewriting prompts. The four generation templates are listed below. Contextual generation returns two differently framed candidates, while each of the remaining templates returns one rewritten problem. Reasoning traces and answers are produced after answer-consistency filtering rather than requested by these rewriting templates.

Listing 1: Contextual-rewriting prompt.

```
# Role
You are a senior expert in mathematical problem rewriting.

# Task
Read the original math problem and generate exactly two contextual
    rewrites.

# Hard Constraints
1. Preserve the exact mathematical core, constraints, solvability, and
    difficulty.
2. Both rewrites must keep the same answer type as the original
    problem.
```

```
3. Both rewrites must be deterministic and exam-ready.
4. The two rewrites must use different real-world domains and
    noticeably different narrative framing.
5. Do not add unnecessary background or hidden assumptions.
6. Keep formulas, definitions, and conditions explicit enough that the
    math structure stays clear.

# Output Format
Return only one raw JSON object:
{
  "contextual_rewrite_1": "full rewritten problem text",
  "contextual_rewrite_2": "full rewritten problem text"
}

# Original Problem
<<<PROBLEM>>>
```

Listing 2: Synonym-rewriting prompt.

```
You are a senior mathematics assessment editor.
Rewrite the following math problem with minimal surface change only.

Requirements:
1. Preserve all mathematical meaning, symbols, quantities, and
    constraints.
2. Do not change the answer.
3. Only perform light synonym or sentence-pattern rewriting.
4. Keep the problem deterministic and exam-ready.
5. Do not add extra story, hints, or explanations.

Return only one raw JSON object:
{
  "augmented_query": "rewritten problem text"
}

Problem:
<<<PROBLEM>>>
```

Listing 3: Variable-substitution prompt.

```
You are an experienced mathematics editor.
Perform variable substitution on the following problem.

Requirements:
1. Rename free variables consistently.
2. Do not rename constants, functions, operators, or units.
3. Do not change numerical values or mathematical relationships.
4. Keep the problem deterministic and exam-ready.

Return only one raw JSON object:
{
  "augmented_query": "rewritten problem text"
}

Problem:
<<<PROBLEM>>>
```

Listing 4: Numerical-substitution prompt.

```
You are a mathematics item writer.
Create a numerically substituted variant of the following problem.
```

```
Requirements:
1. Preserve the same mathematical structure and difficulty level.
2. Keep the rewritten problem mathematically valid and determinate.
3. If the original numbers have special properties, preserve those
    properties.
4. Update any answer choices if the problem is multiple choice.
5. Keep the problem exam-ready.

Return only one raw JSON object:
{
  "augmented_query": "rewritten problem text"
}

Problem:
<<<PROBLEM>>>
```

A.1 Problem-Rewriting Comparisons

Table 5 presents three context-rewriting pairs. The pairs preserve, respectively, the allowed swaps and reachability objective, the subset sizes and pairwise intersection constraint, and the factorial trailing-zero equation. Each column traces one problem to its meta-problem and concept list, aligned as in Table 3. Bold text marks corresponding constraints, objectives, and concepts.

B Additional Experimental Results

Unless otherwise specified, the appendix experiments are conducted on the *aops_forum* evaluation set with the same retrieval pool, prompt template, decoding configuration, and answer-equivalence judge as the main experiments. All tables in this section are single-column tables. A denotes the uncontrolled split, B denotes the matched split, and ALL denotes their union.

B.1 Retrieval of Constructed Rewrites

For each query $x \in \mathcal{T}_B$, where $\mathcal{T}_B$ is the matched test subset, let $\mathcal{R}(x) \subseteq \mathcal{D}$ denote the rewrites constructed from x and included in the demonstration pool, and let $\mathcal{Z}_3(x)$ be the evaluated retriever's top-three results. We define

$$\text{Hit@3} = \frac{1}{|\mathcal{T}_B|} \sum_{x \in \mathcal{T}_B} \mathbb{I}[\mathcal{Z}_3(x) \cap \mathcal{R}(x) \neq \varnothing]\,. \tag{12}$$

Each query contributes one hit if at least one of its constructed rewrites is retrieved, regardless of how many appear. Table 6 reports this metric as a percentage. The metric evaluates recovery of the controlled rewrite pairs only; other retrieved demonstrations may still provide useful solution guidance.

B.2 Reward-Parameter Ablation

The GRPO reward compares zero-shot correctness z and retrieved few-shot correctness f. The two tunable entries are r_{11}, which corresponds to the case where both zero-shot and few-shot answers are correct, and r_{00}, which corresponds to the case where both answers are incorrect. The default setting used by MetaPR is $r_{11} = 0.05$ and $r_{00} = -0.10$. Table 7 merges the two one-variable sweeps: the first block fixes r_{00} and varies r_{11}, while the second block fixes r_{11} and varies r_{00}. The reported columns are the ALL accuracies of the three solving models on *aops_forum*.

The default setting provides the strongest average accuracy across the three solvers, although nearby values yield the best result for individual models. Qwen2.5-7B-Instruct favors $r_{11} = 0.00$ and $r_{00} = -0.05$, whereas Llama-3.2-3B-Instruct peaks at $r_{11} = -0.05$ and $r_{00} = 0.00$. The non-monotonic response suggests that the reward values primarily control a trade-off between preserving already-correct predictions and encouraging retrieval-induced corrections.

B.3 Feature-View Ablation

Table 8 studies which retrieval views are used to construct the query representation. Here x denotes the original problem, m denotes the generated *meta-problem*, and c denotes the concept list. Each

Table 5: Context rewrites with preserved mathematical constraints and objectives: seating permutations, intersecting set families, and factorial trailing zeros. Problem statements are condensed for display while retaining their mathematical conditions and objectives.

(a) Allowed swaps and graph connectivity

Original	Rewritten
n people (named $1, 2, \ldots, n$) sit around a table. Some of them are friends. At each step, **two friends may exchange places**. Find a necessary and sufficient condition on the friendship relation such that these steps can reach **every possible permutation**.	At a roundtable conference, n delegates (numbered $1, 2, \ldots, n$) are seated. **Any two acquainted delegates may swap seats**. Find a necessary and sufficient condition on the acquaintance relation for reaching **every possible seating arrangement** through such swaps.
meta_problem Given a friendship relation on a set of n people, find a **necessary and sufficient condition** for the relation that allows reaching **any permutation** of their seating arrangement through **pairwise swaps**.	**meta_problem** Given a graph with n vertices and an acquaintance relation represented as edges, determine a **necessary and sufficient condition** on the graph such that **any permutation** of the vertices can be achieved through a sequence of **edge swaps**.
concept_list **graph theory**; **permutations**; **graph connectivity**; graph isomorphism; pairwise swaps	**concept_list** **graph theory**; **permutations**; **graph connectivity**; graph isomorphism; swap operations

(b) Intersecting subsets and botanical samples

Original	Rewritten
What is the **maximum number of subsets of size 5**, taken from the set $A = \{1, 2, 3, \ldots, 20\}$, such that **any two share exactly one element**?	A botanical lab chooses 5-plant samples from 20 distinct species numbered 1 through 20. **Any two samples must share exactly one common plant species**. What is the **maximum number of such 5-plant samples**?
meta_problem Given a set and a subset size, find the **maximum number of subsets** such that **any two subsets intersect in exactly one element**.	**meta_problem** Given a set of 20 distinct elements, find the **maximum number of 5-element subsets** such that **any two subsets intersect in exactly one element**.
concept_list **combinatorial design**; **block design**; **intersection properties**; Pigeonhole Principle	**concept_list** **combinatorial design**; **block design**; **intersection properties**; finite projective planes

(c) Factorial trailing zeros and program scores

Original	Rewritten
Find the integer $n \geq 48$ for which the **number of trailing zeros** in the decimal representation of $\boldsymbol{n!}$ is **exactly** $\boldsymbol{n - 48}$.	A competition program outputs $n!$ in decimal for $n \geq 48$. Its score is the **number of trailing zeros** in that output. Find n if the score is **exactly** $\boldsymbol{n - 48}$.
meta_problem Given an integer $n \geq 48$, find the integer n such that the **number of trailing zeros in $\boldsymbol{n!}$** is exactly $n - 48$.	**meta_problem** Given a **factorial expression** and constraints on **trailing zeros**, find the integer satisfying a specific count relation.
concept_list **trailing zeros in factorials**; divisibility by 10; counting factors of 5 and 2; asymptotic analysis of factorials	**concept_list** Legendre's formula; **factorial trailing zeros**; prime factorization; exponential properties; number theory

Table 6: Retrieval of constructed rewrites on the matched split B. Hit@3 measures whether the top three results contain at least one rewrite constructed from the query. Best results are shown in bold.

Method	*aops_forum* Hit@3	*olympiads* Hit@3
BM25	22.08%	23.77%
Dense	28.75%	30.74%
Skill-KNN	30.00%	31.56%
GraphIC	37.92%	39.34%
RGER	34.17%	36.48%
MetaPR (Base)	27.92%	29.92%
MetaPR (SFT)	36.67%	38.52%
MetaPR (GRPO)	**42.08%**	**44.26%**

Table 7: Reward-parameter ablation on *aops_forum*. The first block varies r_{11} with $r_{00} = -0.10$, while the second varies r_{00} with $r_{11} = 0.05$. Best results within each sweep are bold.

Fixed Reward	Varied Reward	*aops_forum* ALL Acc.		
		Llama-3.1-8B Instruct	Qwen2.5-7B Instruct	Llama-3.2-3B Instruct
$r_{00} = -0.10$	$r_{11} = -0.10$	33.38%	30.25%	26.50%
	$r_{11} = -0.05$	33.88%	29.88%	**27.38%**
	$r_{11} = 0.00$	33.63%	**31.00%**	26.63%
	$r_{11} = 0.05$	**34.25%**	30.88%	27.25%
	$r_{11} = 0.10$	33.75%	30.38%	26.75%
$r_{11} = 0.05$	$r_{00} = -0.10$	**34.25%**	30.88%	27.25%
	$r_{00} = -0.05$	34.00%	**31.13%**	26.88%
	$r_{00} = 0.00$	33.50%	30.50%	**27.50%**
	$r_{00} = 0.05$	33.75%	30.00%	26.63%
	$r_{00} = 0.10$	33.00%	29.75%	26.13%

row specifies a view combination and reports the ALL accuracy of the three solving models on *aops_forum*.

Table 8: Feature-view ablation on *aops_forum*. The views x, m, and c denote the original problem, generated *meta-problem*, and concept list, respectively. Best results are bold.

Views	Llama-3.1-8B Instruct	Qwen2.5-7B Instruct	Llama-3.2-3B Instruct
x	24.75%	22.38%	18.38%
m	27.75%	26.25%	20.00%
c	24.13%	23.38%	16.88%
$x + m$	31.75%	28.75%	24.38%
$x + c$	29.50%	27.88%	22.75%
$m + c$	30.63%	29.13%	25.00%
$x + m + c$	**34.25%**	**30.88%**	**27.25%**

The full representation exceeds every single-view and two-view alternative, with gains of 2.50, 1.75, and 2.25 points over the strongest reduced variant for Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Llama-3.2-3B-Instruct, respectively. This result indicates that the three views provide complementary retrieval information.

B.4 Generator-Backbone Comparison

Table 9 compares abstraction generators while keeping the downstream solver fixed. The experiment is conducted only on *aops_forum*. The second column identifies the abstraction source and training stage; "No retrieval" denotes zero-shot inference.

The GRPO-trained generator yields the best result for all three solvers. Compared with Qwen2.5-7B-Instruct (Base), Qwen2.5-7B-Instruct (GRPO) improves ALL accuracy by 7.00 points for

Table 9: Comparison of abstraction generators on *aops_forum*. "Base," "SFT," and "GRPO" identify the training stage of Qwen2.5-7B-Instruct. Best results within each solver block are bold.

Solver	Abstraction Generator	*aops_forum* Acc.		
		A	B	ALL
Llama-3.1-8B Instruct	No retrieval	11.79%	5.83%	10.00%
	Qwen2.5-7B-Instruct (Base)	21.96%	39.58%	27.25%
	Llama-3.1-8B-Instruct	24.11%	44.17%	30.13%
	GLM-4-9B-0414	23.21%	42.92%	29.13%
	Qwen2.5-7B-Instruct (SFT)	**29.11%**	38.33%	31.88%
	Qwen2.5-7B-Instruct (GRPO)	28.93%	**46.67%**	**34.25%**
Qwen2.5-7B Instruct	No retrieval	24.46%	20.83%	23.38%
	Qwen2.5-7B-Instruct (Base)	24.11%	22.08%	23.50%
	Llama-3.1-8B-Instruct	25.54%	23.75%	25.00%
	GLM-4-9B-0414	26.07%	26.67%	26.25%
	Qwen2.5-7B-Instruct (SFT)	25.36%	**34.17%**	28.00%
	Qwen2.5-7B-Instruct (GRPO)	**29.64%**	33.75%	**30.88%**
Llama-3.2-3B Instruct	No retrieval	11.61%	8.33%	10.63%
	Qwen2.5-7B-Instruct (Base)	15.18%	30.42%	19.75%
	Llama-3.1-8B-Instruct	15.89%	35.00%	21.63%
	GLM-4-9B-0414	16.43%	35.83%	22.25%
	Qwen2.5-7B-Instruct (SFT)	**22.14%**	30.42%	24.63%
	Qwen2.5-7B-Instruct (GRPO)	20.89%	**42.08%**	**27.25%**

Llama-3.1-8B-Instruct, 7.38 points for Qwen2.5-7B-Instruct, and 7.50 points for Llama-3.2-3B-Instruct. The relative ordering of Llama-3.1-8B-Instruct and GLM-4-9B-0414 as generators varies across solvers, indicating that backbone scale alone does not determine retrieval quality.

B.5 Effect of the Number of Retrieved Demonstrations

Table 10 studies how the number of retrieved demonstrations affects final-answer accuracy on *aops_forum*. We vary k from 2 to 6 while keeping the retriever, prompt format, generator checkpoint, and solver decoding configuration fixed.

Table 10: Effect of the number of retrieved demonstrations k on *aops_forum*. Columns report ALL accuracy. The marker † denotes the shared main-experiment setting, and the best result for each solver is bold.

k	Llama-3.1-8B Instruct	Qwen2.5-7B Instruct	Llama-3.2-3B Instruct
2	33.38%	30.00%	**27.50%**
3†	**34.25%**	30.88%	27.25%
4	33.88%	**31.13%**	27.00%
5	33.13%	30.63%	26.25%
6	33.50%	30.13%	26.50%

The shared setting $k = 3$ provides the strongest average accuracy across the three solvers. Llama-3.1-8B-Instruct peaks at $k = 3$, while Qwen2.5-7B-Instruct and Llama-3.2-3B-Instruct obtain slightly higher accuracy at $k = 4$ and $k = 2$, respectively. The variation across solvers suggests that the optimal context size depends on model capacity and sensitivity to additional demonstrations rather than increasing uniformly with k.

C Prompts and Implementation Details

C.1 Training Configuration

The Qwen2.5-7B-Instruct generator is trained with LoRA rank 8, alpha 16, and dropout 0.05. SFT uses a learning rate of 2×10^{-5} for three epochs. GRPO uses the same LoRA configuration, a learning rate of 10^{-6} for one epoch, eight generations per group, a KL coefficient of 0.04, and a clipping

value of 0.2. The frozen reward-solver ensemble $\mathcal{M}_{\mathrm{RL}}$ comprises Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, and GLM-4-9B-0414; the per-solver correctness-transition rewards are summed as shown in Eq. (9). Retrieval uses three equally weighted BGE-M3 views with $k = 3$. Demonstration-side abstractions remain fixed during GRPO, and all experiments use seed 42.

C.2 Auxiliary Reward Evaluation

The generator serializes the meta-problem m and concept list c as `superordinate_problem` and `key_concepts`, respectively. These fields define retrieval views; the downstream solver continues to receive the original target problem. Let $s_j^{(g)}$ denote the raw generator output and let $\mathcal{J}_{\mathcal{S}}$ contain the strings that are directly parseable as JSON objects with exactly these two fields. The first field must be a nonempty string, and the second must be a nonempty array of nonempty strings. The format reward is

$$r_{\mathrm{fmt},j}^{(g)} = \mathbb{I}\Big[s_j^{(g)} \in \mathcal{J}_{\mathcal{S}}\Big]. \tag{13}$$

Responses containing additional fields, surrounding text, Markdown code fences, empty values, or incompatible value types receive no format reward.

For the soft length penalty, define

$$\psi(v; L) = \min\left(1, \frac{\max(0, v - L)}{L}\right). \tag{14}$$

Let $\ell(\cdot)$ count Python characters after field extraction, $|c|$ denote the number of concepts, and $\ell_{\mathrm{extra}}(s)$ count normalized characters outside the extracted JSON object. We use

$$\begin{aligned} p_{\mathrm{len},j}^{(g)} = {} & 0.15\,\psi(\ell(m_j^{(g)}); 150) + 0.10\,\psi(|c_j^{(g)}|; 10) \\ & + 0.15 \sum_{u \in c_j^{(g)}} \psi(\ell(u); 30) + 0.20\,\psi(\ell_{\mathrm{extra}}(s_j^{(g)}); 40). \end{aligned} \tag{15}$$

The complete penalty is multiplied by $\lambda_{\mathrm{len}} = 0.5$ in Eq. (9). These thresholds are soft budgets rather than rejection or truncation rules.

C.3 Abstraction Prompt

Listing 5 gives the two-field abstraction instruction. The original problem is appended after the final line.

Listing 5: Abstraction-generation prompt.

```
[ROLE]
You are a top-tier mathematics competition coach with expertise in
    identifying the core mathematical paradigm of complex problems.
    Your task is to extract the mathematical method(s), abstract
    target, and known conditions from a given Original Problem,
    ignoring all contextual and narrative elements, and express them
    in a highly abstract and formalized manner. Additionally, identify
    and output the key theorems, formulas, concepts, and principles
    involved in solving the problem.

[OBJECTIVE]
For each Original Problem, convert it into a Superordinate Problem and
    identify the key mathematical concepts involved, in the following
    form:

1. Superordinate Problem:
   - Identify the essential known conditions (e.g., equations,
    inequalities, geometric conditions, number constraints, etc.).
    These should be abstractly generalized to a mathematical model
    (e.g., "given two linear equations", "given a geometric
    configuration", etc.).
   - Express the target of the problem in its most abstract form
    (e.g., solve for the intersection point, find the minimum/maximum
    of a function, determine the solution set).
```

```
2. Key Concepts:
   - Identify the core theorems, formulas, methods, and principles
    involved in solving the problem (e.g., "Pythagorean Theorem",
    "Cauchy-Schwarz Inequality", "modular arithmetic", "divisibility
    rules", "system of linear equations", etc.). These should be
    listed as individual words or short phrases.

Ensure high-level abstraction: The statement should clearly highlight
    the mathematical tools and paradigms necessary for solving the
    problem while remaining precise and understandable to a
    knowledgeable audience. Avoid irrelevant details and contextual
    explanations.

[OUTPUT FORMAT]
Your output must be a single JSON object with exactly two keys:

{
  "superordinate_problem": "...",
  "key_concepts": [...]
}

Where:
- superordinate_problem is a single sentence of the form:
  "Given <abstract known conditions>, find <abstract target>."
- key_concepts is an array of strings containing the key concepts
    (e.g., theorems, formulas, principles, etc.) used in solving the
    problem. These should be words or short phrases.

[CONVENTIONS FOR ABSTRACT TARGETS]
Use descriptors such as:
  "solve for the intersection point",
  "count the number of sequences satisfying modular conditions",
  "find the minimum/maximum of an expression under constraints",
  "characterize all solutions meeting structural conditions",
  "determine the existence or structure of solutions",
  "prove a relation holds under given constraints".

[Few-Shot Example - Olympiad-Level Abstract Patterns]

Example 1:
Original:
"A telecommunications engineer configures an 11-digit header with
    specific fixed and variable positions drawn from a fixed set of
    digits so that the resulting number is divisible by 15. How many
    such headers exist?"
Superordinate:
{
  "superordinate_problem": "Given a set of digits and divisibility
    constraints, find the number of sequences satisfying modular
    conditions.",
  "key_concepts": ["divisibility rules", "modular arithmetic",
    "combinatorial counting"]
}

Example 2:
Original:
"Find the intersection of two lines in the plane given by their
    equations (ax + by = c) and (dx + ey = f)."
Superordinate:
{
  "superordinate_problem": "Given two linear equations in two
    variables, find the intersection point.",
  "key_concepts": ["system of linear equations", "substitution",
    "elimination"]
```

```
}

Example 3:
Original:
"In triangle ABC, given a segment parallel to the base creates
    proportional subdivisions on two sides, prove the ratios of these
    subdivisions are equal."
Superordinate:
{
  "superordinate_problem": "Given a triangle with a parallel segment
    dividing sides proportionally, find the proportional ratios of the
    subdivisions.",
  "key_concepts": ["parallel line proportionality", "similar
    triangles", "proportionality theorems"]
}

Example 4:
Original:
"Find all functions f: R -> R such that for all real x and y,
    f(x+f(y)) + f(y+f(x)) = 2(x+y)."
Superordinate:
{
  "superordinate_problem": "Given a functional equation with symmetry,
    find all functions satisfying a coupled additive constraint.",
  "key_concepts": ["functional equations", "symmetry", "additive
    constraints"]
}

Example 5:
Original:
"Let n be a positive integer. Prove that for any real numbers
    x1,x2,...,xn with sum 0, x1^3 + x2^3 + ... + xn^3 >= x1x2x3 + ...
    holds under given conditions."
Superordinate:
{
  "superordinate_problem": "Given a set of real numbers with a sum
    condition, find the lower bound of a homogeneous polynomial.",
  "key_concepts": ["sum inequalities", "symmetry", "homogeneous
    polynomials"]
}

Now process the Original Problem below.
```

C.4 Solver and Judge Prompts

The solver receives the original target problem and, in the few-shot condition, the retrieved demonstrations in rank order. Each demonstration contributes its original problem and solution. If a stored demonstration solution does not already begin with the step-by-step cue, the cue is prepended before prompt assembly. The resulting user templates are:

Listing 6: Zero-shot and few-shot solver templates.

```
System:
You are a math problem solver. You should think step by step.

Zero-shot user message:
Question: {target_problem}
Answer:

Few-shot user message:
Question: {demonstration_problem_1}
Answer: {demonstration_solution_1}

...
```

```
Question: {target_problem}
Answer:
```

The same binary answer-equivalence protocol is used for main evaluation and candidate utility evaluation. The reference includes the final answer and, when available, its solution. Only an explicit *Verdict: EQUIVALENT* is counted as correct; a missing model answer, an indeterminate comparison, or an unparseable judge output is assigned *NOT_EQUIVALENT*.

Listing 7: Binary answer-equivalence judge prompt.

```
You are a mathematical answer validator.
Compare the reference and the model output for the same problem.
If the final mathematical answer is equivalent, output EQUIVALENT.
If it is not equivalent, missing, or cannot be determined, output
    NOT_EQUIVALENT.

PROBLEM:
{problem}

REFERENCE:
{reference}

MODEL OUTPUT:
{generation}

Reply in this format:
Reason: <brief reason in 1-2 sentences>
Verdict: EQUIVALENT or NOT_EQUIVALENT
```