# ReasonFlux: Hierarchical LLM Reasoning via Scaling Thought Templates

**Ling Yang** [1 *] **Zhaochen Yu** [2 *] **Bin Cui** [2] **Mengdi Wang** [1]
[1]Princeton University [2]Peking University
Code: https://github.com/Gen-Verse/ReasonFlux

## Abstract

We present that hierarchical LLM reasoning via scaling *thought templates* can effectively optimize the reasoning search space and outperform the mathematical reasoning capabilities of powerful LLMs like OpenAI o1-preview and DeepSeek V3. We train our *ReasonFlux*-32B model with only 8 GPUs and introduces three innovations: (**i**) a structured and generic thought template library, containing **around 500 high-level thought templates** capable of generalizing to similar or relevant reasoning problems; (**ii**) performing hierarchical reinforcement learning on a sequence of thought templates instead of original long CoT data, optimizing a base LLM to plan out an optimal template trajectory for gradually handling complex problems; (**iii**) a brand new inference scaling system that enables *hierarchical LLM reasoning* by adaptively scaling thought templates at inference time. With a template trajectory containing sequential thought templates, our *ReasonFlux*-32B significantly advances math reasoning capabilities to state-of-the-art levels. Notably, on the MATH benchmark, it achieves an accuracy of **91.2%** and surpasses o1-preview by 6.7%. On the USA Math Olympiad (AIME) benchmark, *ReasonFlux*-32B solves an average of **56.7%** of problems, surpassing o1-preview and DeepSeek-V3 by 27% and 45%, respectively.

| Task | **ReasonFlux 32B** | **DeepSeek V3** | **OpenAI o1-preview** | **OpenAI o1-mini** | **QWQ 32B-preview** | **GPT 4o** |
|---|---|---|---|---|---|---|
| MATH | **91.2** | 90.2 | 85.5 | 90.0 | 90.6 | 76.6 |
| AIME 2024 | **56.7** | 39.2 | 44.6 | 56.7 | 50.0 | 9.3 |
| Olympiad Bench | **63.3** | 55.4 | - | 65.3 | 61.2 | 43.3 |
| GaokaoEn 2023 | **83.6** | - | 71.4 | 78.4 | 65.3 | 67.5 |
| AMC2023 | **85.0** | 80.0 | 90.0 | 95.0 | - | 47.5 |

*Table 1.* Performance Comparison on Various Math Reasoning Benchmarks (Pass@1 Accuracy)

## 1. Introduction

Large Language Models (LLMs) have recently achieved remarkable progress, demonstrating exceptional capabilities in tackling complex reasoning tasks and even surpassing human experts in specific domains. For example, models such as OpenAI's O1 (Jaech et al., 2024), Google's Gemini-2.0 (Team et al., 2024), DeepSeek-V3 (Liu et al., 2024b), and Qwen-QwQ (Team, 2024a) are at the forefront of this progress, characterized by their ability to emulate human reasoning through a slower, more deliberate thought process. These models leverage increased inference time to enhance reasoning accuracy. While they have unlocked substantial performance gains, more complex tasks such as mathematical problem solving in AIME, OlympiadBench (He et al., 2024) and code in LiveCodeBench (Jain et al., 2024), which demand a more fine-grained search through a vast solution space and more delicate thought for each intricate reasoning step, thus still pose significant challenges.

Subsequent research has focused on enhancing LLMs' reasoning capabilities on complex problems through inference-time

[*]Equal contribution . Correspondence to: Ling Yang <yangling0818@163.com>, Mengdi Wang <mengdiw@princeton.edu>.

Preprint.

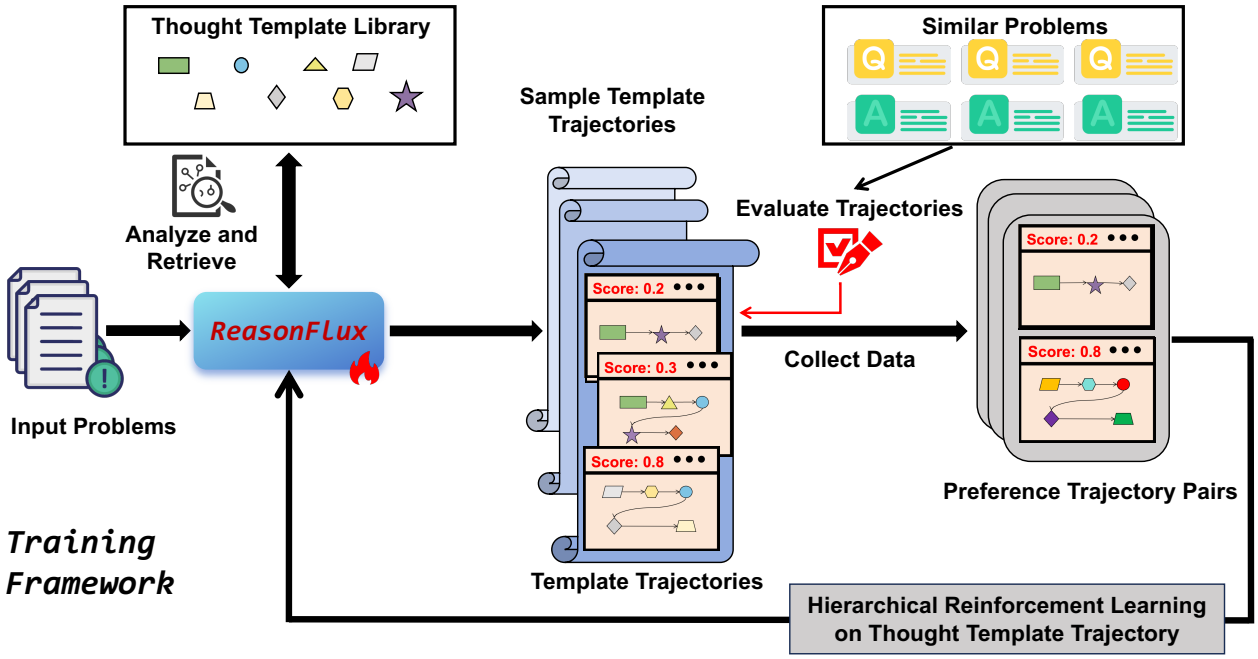


*Figure 1.* **Training framework for our *ReasonFlux*.** We train with hierarchical reinforcement learning to enable the model to plan out an optimal and generalizable thought template trajectory for an input problem. Our new inference-scaling framework is in Figure 2.

strategies. These strategies can be divided into two categories: deliberate search and reward-model-guided methods. Deliberate search methods, like Tree of Thoughts (ToT) (Yao et al., 2024) and Graph of Thoughts (GoT) (Besta et al., 2024), allow LLMs to explore multiple reasoning paths and self-evaluate choices to find the optimal trajectory. Reward-model-guided methods leverage reward models to assess reasoning step quality. Best-of-N approaches, which leverage an Outcome Reward Model (ORM) to find the optimal reasoning paths in multiple candidates, while Process Reward Models (PRMs) (Lightman et al., 2023; Luo et al., 2024; Wang et al., 2024) guide the model towards promising paths by rewarding high-probability intermediate steps. Building on this, Monte Carlo Tree Search (MCTS) (Zhang et al., 2024a; Qi et al., 2024) employs a fine-grained search, decomposing tasks into simpler steps and using PRMs to guide action selection within a tree-based search space. However, these methods often incur high computational costs, especially with numerous reasoning steps or vast search spaces, primarily due to the inherent randomness of sampling, which hinders the efficient identification of the optimal reasoning trajectory. Furthermore, they rely on manually designed search strategies and instance/step-level reward, limiting their generalization ability to diverse and complex reasoning tasks. Essentially, they struggle to effectively balance the exploration-exploitation trade-off during inference scaling. This highlights the need for a more efficient and generalizable inference scaling approach that enhances reasoning without extensive manual effort, while providing a more principled search strategy.

To achieve more efficient and precise search of reasoning paths, a feasible approach is to utilize Retrieval-Augmented Generation (RAG). Recent Buffer of Thought (BoT) (Yang et al., 2024b) constructs a meta-buffer to store informative, high-level thoughts distilled from various problem-solving processes, adaptively retrieving and instantiating relevant thought templates for each specific task. SuperCorrect (Yang et al., 2024c) further utilizes both high-level and detailed thought templates to enhance reasoning ability of small LLMs. Despite significant improvements, such template-based reasoning methods may still face challenges when applied to complex reasoning tasks. Because complex problems often require the integration of multiple templates or diverse pieces of retrieved information, which current methods struggle to address effectively.

To this end, we introduce *ReasonFlux*, a novel hierarchical LLM reasoning framework that configures optimal *thought template trajectories* by automatically retrieving relevant high-level thought templates at inference time, to achieve superior performance on complex reasoning tasks and even **outperform OpenAI o1-preview and o1-mini models**. To be more specific, we first construct a structured template library, which contains **500 useful compacted thought templates** for

efficient retrieval and adaptation. Instead of optimizing a long CoT trajectory, we perform hierarchical reinforcement learning on a sequence of high-level thought templates, optimizing a base llm to learn an optimal **thought template trajectory** from multiple ones and guiding an inference LLM to solve a series of simpler sub-problems. Finally, we develop a new inference scaling system through adaptively **scaling thought templates**. This hierarchical reasoning paradigm enables *ReasonFlux* to simplify the search of reasoning paths and enhance the reasoning ability for complex problems by dynamically selecting a most appropriate high-level template for each sub-problem. Our automated template scaling allows *ReasonFlux* to effectively achieve a better exploration-exploitation trade-off, leading to a more robust and efficient problem-solving process. Through these innovations, *ReasonFlux* offers a more efficient, generalizable, and scalable solution for enhancing the complex reasoning capabilities of LLMs. Finally, we summarize our contributions as follows:

1. We introduce *ReasonFlux* (in Figure 1), a hierarchical LLM reasoning framework that significantly enhances complex reasoning capabilities, outperforming SOTA models like o1-preview and DeepSeek-V3 on challenging MATH and AIME benchmarks (in Table 2).

2. We propose a structured and compact template library with around 500 thought templates curated from challenging mathematical problems. This library facilitates efficient retrieval and adaptation of relevant high-level thought templates for a series detailed reasoning steps.

3. We develop hierarchical reinforcement learning on a sequence of high-level thought templates, to enable LLMs to generate an optimal *thought template trajectory* for a series of simpler sub-problems, effectively simplifying the search space of reasoning paths.

4. We design an new inference scaling system (in Figure 2) by adaptively scaling thought templates for hierarchical reasoning. This system allows *ReasonFlux* to dynamically retrieve a series of high-level templates and adaptively perform instantiated reasoning at inference time, achieving a better exploration-exploitation trade-off for robust and efficient problem-solving.

## 2. Related Work and Discussions

**Learning from Preferences for Language Models** Preference learning is critical for aligning Large Language Models (LLMs) with human expectations and perceptions. Initial approaches, building on pre-training and supervised fine-tuning (SFT), employed PPO in Reinforcement Learning from Human/AI Feedback (RLHF/RLAIF) frameworks (Schulman et al., 2017; Christiano et al., 2017; Ouyang et al., 2022; Xie et al., 2024). These approaches typically involve training a reward model on preference pairs and subsequently optimizing the LLM to maximize the learned reward. However, PPO's instability and inefficiency motivated alternative approaches like DPO (Rafailov et al., 2024), which directly optimizes a policy from paired preference data. Subsequent research has addressed various challenges. ORPO (Hong et al., 2024) integrates alignment into SFT, KTO (Ethayarajh et al., 2024) leverages pointwise data, simplifying data acquisition process. Other efforts focus on finer-grained optimization, such as Step-DPO (Lai et al., 2024) and Cross-DPO (Yang et al., 2024c) that targets intermediate reasoning or reflection steps. SPO (Swamy et al., 2024) employs game-theoretic concepts to address non-transitive preferences, while Multi-turn DPO (Shi et al., 2024) extends optimization to conversations. However, existing methods often rely on instance or step-level reward units, potentially failing to capture and reward the higher-level cognitive processes inherent in human problem-solving process. To this end, we introduce hierarchical RL-based optimization, a novel preference learning approach that encourages the model to configure a series of high-level thought templates that can handle diverse sub-tasks for complex problems, thereby promoting more human-like problem-solving strategies in LLMs.

**Retrieval-Augmented Generation for Language Models** Retrieval-augmented Language Models (RALMs) have become a powerful approach to mitigating hallucinations and enhancing the factual accuracy of LLMs (Asai et al., 2023; Mialon et al., 2023; Shi et al., 2023; Gao et al., 2023; Zhao et al., 2024). By retrieving relevant documents from a large-scale external knowledge source (Borgeaud et al., 2022) to inform response generation, RALMs have demonstrated superior performance in question-answering, often with fewer parameters than traditional LLMs (Mialon et al., 2023). Their versatility is further evidenced by successful applications across diverse tasks, including multi-modal generation and biomedical applications (Yasunaga et al., 2023; Izacard et al., 2023; Wang et al., 2022; Zhao et al., 2024; Borgeaud et al., 2022; Yang et al., 2023). However, RALMs face challenges in complex reasoning tasks, such as math and code, where retrieving relevant guidelines or templates via standard embedding similarity search proves difficult. While methods like RAFT (Zhang et al., 2024c) have attempted to address this by improving retrieval relevance, respectively, their effectiveness decrease as the document

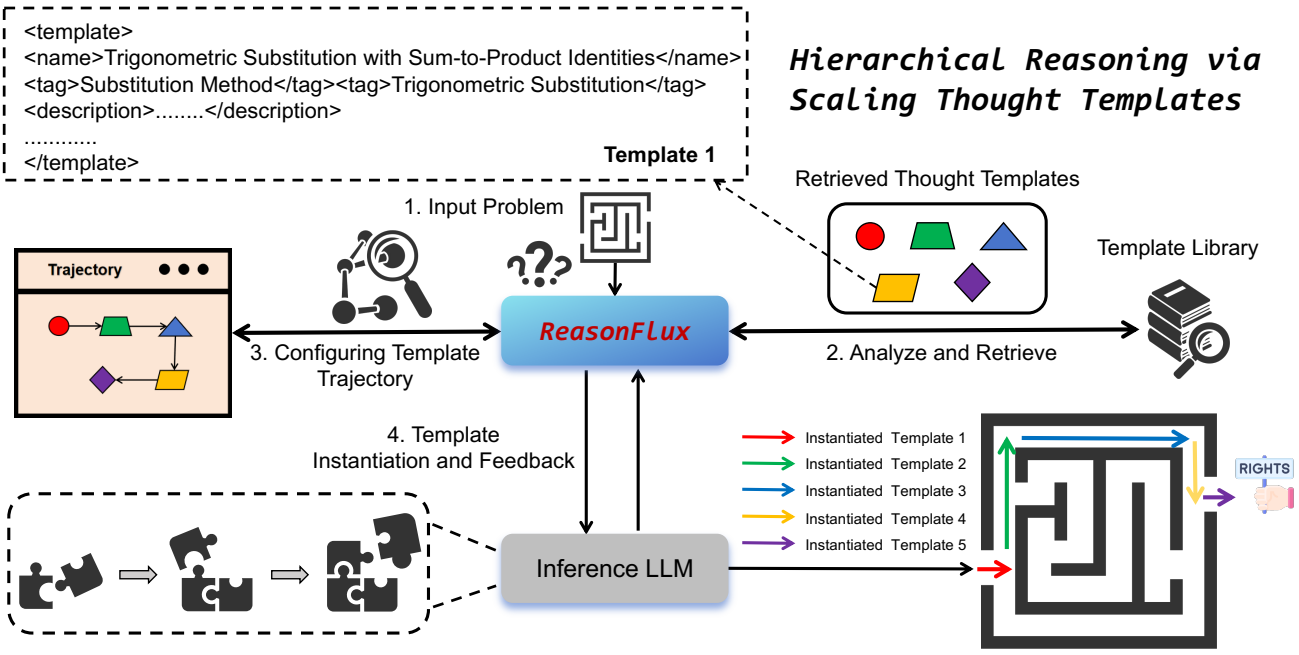


*Figure 2.* **New inference scaling system based on hierarchical reasoning.** We retrieve a series of high-level thought templates for complex problems, and gradually conduct instantiated reasoning for a sequence of sub-problems.

size grows. To overcome these limitations, we design a structured and compact template library for efficient and accurate retrieval, specifically targeting complex reasoning problems.

**Inference Scaling for LLM Reasoning** The auto-regressive nature of LLMs suggests that solving more complex problems inherently requires generating more tokens. Early work, such as CoT (Wei et al., 2022), used prompting techniques like "Let's think step by step" to break down complex reasoning tasks into simpler sub-problems, thus enhancing reasoning performance. Building on this, ToT (Yao et al., 2024) and GoT (Besta et al., 2024) employed different data structures to expand the reasoning space, allowing LLMs to explore multiple solution paths. Recent research (Wu et al., 2024; Snell et al., 2024) has formalized the concept of inference scaling laws, which examine the trade-offs between the generation of additional tokens, and the use of various inference strategies. For instance, majority voting and best-of-N methods (Wang et al., 2023; Li et al., 2023) generate multiple candidate solutions and select the best based on frequency among all the results or the reward model's evaluation. Similarly, approaches using Monte Carlo Tree Search (MCTS) (Zhang et al., 2024b; Liu et al., 2024c; Choi et al., 2023; Zhou et al., 2023) leverage greater search and computation to improve accuracy. To enhance search accuracy, Process Reward Models (PRMs) have been introduced to select high-quality reasoning paths, with studies (Setlur et al., 2024; Snell et al., 2024; Lightman et al., 2023; Luo et al., 2024; Wang et al., 2024) demonstrating their effectiveness, particularly in complex reasoning tasks. More recently, methods like BoT (Yang et al., 2024b) utilize thought templates from past reasoning processes to guide exploration, significantly improving efficiency. However, a deeper understanding of the exploration-exploitation trade-off (Tang et al., 2024; Setlur et al., 2024) for these template-based approaches remains an open challenge. Our work addresses this challenge by scaling an hierarchical template-augmented reasoning paradigm that significantly enhances reasoning accuracy, especially for complex tasks, while strategically balancing exploration and exploitation.

## 3. *ReasonFlux*: Scaling Thought Templates for Hierarchical LLM Reasoning

### 3.1. Constructing Structured Thought Template Library

Inspired by how humans utilize external resources when tackling complex reasoning problems, RAG methods enhance LLMs by enabling them to retrieve information from external sources (Zhao et al., 2024). Recent Buffer of Thought (BoT) (Yang et al., 2024b) attempts to create a buffer of high-level thoughts for llm reasoning, and builds an efficient RAG reasoning system. Despite a comprehensive template library to solve similar problems, BoT still faces scalability challenges

as template size grows, same as the traditional RAG systems that rely on embedding similarity to search unstructured text corpora.

To address this, our approach focuses on constructing a **structured thought template library** that enables more precise, targeted retrieval and mitigates scalability challenges. To build this library, we carefully selected a wide and diverse range of challenging mathematical reasoning problems from different sources, ensuring robustness and broad applicability of our template library. We used an LLM to analyze the thought behind the solution and generating concise summaries of problem-solving strategies and identifying common patterns. This process yielded a collection of high-quality, solution-oriented thought templates. Each template $T_i$ in the library is structured for efficient retrieval and application, where $T_{\text{nam}}$ is the **name** (e.g., "$\sqrt{R^2 - x^2}$ Type Trigonometric Substitution"), $T_{\text{tag}}$ is a set of **tags** for keyword-based retrieval (e.g., {"Trigonometric Substitution", "Irrational Function Optimization"}), $T_{\text{des}}$ is a **description** of the underlying principle and applicable scenarios, $T_{\text{sco}}$ defines the **scope**, specifying the problem types it addresses, $T_a$ is a sequence of detailed **application steps** $\{a_1, a_2, ..., a_k\}$, and $T_{\text{exa}}$ is a set of **examples** demonstrating its application. The entire library $\mathcal{D}_{\text{temp}}$ is a set of thought templates as mentioned:

$$\mathcal{D}_{\text{temp}} = \{T_1, T_2, ..., T_m\} \tag{1}$$

where $m$ is the total number of templates. Here we present an illustration of a thought template within our library. For the sake of brevity, some fields in the following example have been simplified. Please refer to Appendix A for more detailed examples.

**Example Template**

**name**: $\sqrt{R^2 - x^2}$ Type Trigonometric Substitution
**tag**: Substitution Method, Trigonometric Substitution, Irrational Function
**description**: When a radical of the form $\sqrt{R^2 - x^2}$ appears in a problem, and $|x| \leq R$, consider using trigonometric substitution $x = R\sin\theta$ or $x = R\cos\theta$ to eliminate the radical, converting the irrational expression into a trigonometric expression. This allows simplification and problem-solving using the properties and identities of trigonometric functions.
**scope**: Problems involving function optimization or range, especially those involving irrational functions of the form $\sqrt{R^2 - x^2}$. Equations or inequalities containing radicals of the form $\sqrt{R^2 - x^2}$. Geometric problems related to circles.
**application steps**:
1. **Determine the range**: Based on the problem conditions, determine the range of $x$, usually $|x| \leq R$.
... (Steps 2-5 omitted for brevity)
**example**:
... (Examples omitted for brevity)

Efficient retrieval is facilitated by leveraging the metadata associated with each template, specifically the **name** ($n$) and **tags** ($t$), enabling quick and accurate searching based on keywords or specific problem characteristics. This structured organization, combined with rich metadata, ensures that the most relevant templates are readily available for any given problems.

### 3.2. Hierarchical Reinforcement Learning on Thought Template Trajectory

While our structured template library provides a valuable resource for reasoning, an effective method is needed to utilize this library and select the appropriate templates for handing a given problem. To this end, we perform hierarchical reinforcement learning to train and finally obtain ***ReasonFlux*** that can effectively plan out an optimal *thought template trajectory* for a problem. We retrieve and configure a sequence of relevant templates from the library, assisting in instantiating the retrieved templates on specific sub-problems. *ReasonFlux* acts as an experienced navigator, providing the optimal trajectory denoted as $\mathbb{T}_{\text{traj}}$ that enabling the LLM to instantiate abstract thought templates into concrete sequential problem-solving steps.

**Structure-based Finetuning** Our hierarchical RL process begins by leveraging the structured template library $\mathcal{D}_{\text{temp}}$ to construct a knowledge-intensive training dataset $\mathcal{D}_{\text{train}}$. This dataset comprises diverse examples of template names $T_{\text{nam}}$, their associated tags $T_{\text{tag}}$, detailed descriptions of their underlying principles $T_{\text{des}}$, and a clear delineation of their applicable scopes $T_{\text{sco}}$, represented as tuples $(T_{\text{nam}}, T_{\text{tag}}, T_{\text{des}}, T_{\text{sco}})$ extracted from $\mathcal{D}_{\text{temp}}$. We then fine-tune a base LLM, denoted as $\pi$, on this dataset $\mathcal{D}_{\text{train}}$. This process equips the model with a foundational understanding of the structure, content, and

intended use of each template within the library. The fine-tuning process is driven by the following optimization objective:

$$\mathcal{L}_{\text{struct}} = -\mathbb{E}_{\mathcal{D}_{\text{train}}} \left[ \log \pi(T_{\text{des}}, T_{\text{sco}} | T_{\text{nam}}, T_{\text{tag}}) \right], \tag{2}$$

where the objective is to maximize the likelihood of the model generating the correct description $T_{\text{des}}$ and scope $T_{\text{sco}}$ given the template name $T_{\text{nam}}$ and tags $T_{\text{tag}}$. This ensures that the fine-tuned model can effectively associate the identifying information ($T_{\text{nam}}$ and $T_{\text{tag}}$) of a template with its functional aspects ($T_{\text{des}}$ and $T_{\text{sco}}$). After fine-tuning, we denote the resulting model as $\pi_{\text{struct}}$.

**Preference Learning on Thought Template Trajectory** Based on the finetuned LLM $\pi_{\text{struct}}$, we can further enhance its ability to plan out a sequence of high-level thought templates (*i.e.*, **thought template trajectory** $\mathbb{T}_{\text{traj}}$) for an input problem $x$, associating each step with the most relevant template from the library. This is achieved through our preference learning on thought template trajectory. Specifically, as shown in Figure 1, given an input problem $x$, $\pi_{struct}$ first analyzes and abstracts the problem's conditional information, identifying the core mathematical concepts and relationships involved. Based on this abstract representation, the navigator $\pi_{\text{struct}}$ then configures a trajectory $\mathbb{T}_{\text{traj}} = \{s_1, s_2, ..., s_n\}$, where each $s_i$ represents a high-level step in the reasoning process, associated with a specific template name retrieved from the library which could be used to solve the problem, denoted as $T_i$. Each retrieved template $T_i$ is then instantiated with specific details from the input problem $x$ and provides fine-grained guidance to a separate inference LLM denoted as $\pi_{\text{inf}}$ to solve the problem.

To measure the effectiveness and generalization ability of a given trajectory, we utilize a set of problems $\mathcal{X}_{sim}$ that are similar to the original input problem $x$, including $x$ itself. We then use the instantiated templates along the trajectory $\mathbb{T}_{\text{traj}}$ to guide $\pi_{inf}$ in solving each problem $x_i \in \mathcal{X}_{sim}$. The average accuracy achieved by $\pi_{inf}$ across these problems serves as the **trajectory reward** $R(\mathbb{T}_{\text{traj}})$. Formally:

$$R(\mathbb{T}_{\text{traj}}) = \frac{1}{|\mathcal{X}_{sim}|} \sum_{x_i \in \mathcal{X}_{sim}} \text{Acc}(\pi_{inf}(x_i, \mathbb{T}_{\text{traj}})) \tag{3}$$

where $\text{Acc}(\pi_{inf}(x_i, \mathbb{T}_{\text{traj}}))$ represents the accuracy of $\pi_{inf}$ in solving problem $x_i$ when guided by the trajectory $\mathbb{T}_{\text{traj}}$.

This reward signal is then used to construct optimization pairs, enabling us to further refine the navigator $\pi_{struct}$. To be more specific, for each input problem $x$, we sample multiple different $\mathbb{T}_{\text{traj}}$ and evaluate its quality utilizing the template trajectory reward. We define the loss function for optimizing $\pi_{struct}$ as follows:

$$\mathcal{L}_{\text{TTR}}(\theta) = -\mathbb{E}_{(x,(\mathbb{T}^+_{\text{traj}},\mathbb{T}^-_{\text{traj}}))\sim\mathcal{D}_{pair}} \left[ \log \sigma \left( \beta \log \frac{\pi_\theta(\mathbb{T}^+_{\text{traj}}|x)}{\pi_{sft}(\mathbb{T}^+_{\text{traj}}|x)} - \beta \log \frac{\pi_\theta(\mathbb{T}^-_{\text{traj}}|x)}{\pi_{sft}(\mathbb{T}^-_{\text{traj}}|x)} \right) \right] \tag{4}$$

where $\mathcal{D}_{pair}$ is a dataset of optimization pairs. Each pair consists of an input problem $x$ and two trajectories, $\mathbb{T}^+_{\text{traj}}$ and $\mathbb{T}^-_{\text{traj}}$, where $R(\mathbb{T}^+_{\text{traj}}) > R(\mathbb{T}^-_{\text{traj}})$. $\pi_\theta$ represents the the LLM being optimized with parameters $\theta$, initialized from $\pi_{struct}$.

### 3.3. Inference Scaling with Scaling Thought Templates

After hierarchical RL process, we refer to optimized navigator $\pi_\theta$ as *ReasonFlux*. Then, we further design a novel inference scaling system by leveraging automatically planned trajectories and dynamically retrieved thought templates. This system, illustrated in Figure 2, involves a multi-round interplay between the *ReasonFlux*, a structured template library $\mathcal{D}_{\text{temp}}$, and a downstream inference LLM $\pi_{inf}$.

Given an input problem $x$, the first task for *ReasonFlux* is to analyze and extract the core mathematical concepts and relationships embedded within $x$. Based on this abstract representation, denoted as $a(x)$. *ReasonFlux* then configures an optimal template trajectory $\mathbb{T}^*_{\text{traj}}$. This trajectory, represented as a sequence of steps $\mathbb{T}^*_{\text{traj}} = \{s^*_1, s^*_2, ..., s^*_n\}$, is not a rigid, pre-defined path but rather a dynamically generated plan tailored to the specific nuances of the input problem $x$. Each step $s^*_i$ within the trajectory is associated with a specific template name $T_{\text{nam}}$ and $T_{\text{tag}}$ for efficient retrieval. *ReasonFlux* then searches and retrieves a set of most relevant thought templates from the curated thought template library $\mathcal{D}_{\text{temp}}$. Formally,

the retrieval process can be represented as:

$$T_{\text{rag}} = \textit{ReasonFlux}(\{T^i_{\text{nam}}, T^i_{\text{tag}}\}_{i=1}^n, \mathcal{D}_{\text{temp}}), \tag{5}$$

where $T_{\text{rag}} = \{T_1, T_2, ..., T_n\}$ is the set of $n$ retrieved templates that equals to the number of steps in the configured trajectory, and each is a structured template.

Subsequently, based on the $\mathbb{T}^*_{\text{traj}}$ and retrieved templates $T_{\text{rag}}$, *ReasonFlux* will instruct $\pi_{inf}$ to instantiate each steps $s^*_i$ along with corresponding template $T_i$ and problem-specific details from $x$, transforming into concrete instantiated reasoning steps $\hat{s_i}$:

$$\hat{s_i} = \pi_{inf}(x_i, s_i, T_i), \tag{6}$$

where each $\hat{s_i}$ is generated based on the corresponding $s^*_i$, $T_i$, and $x$.

The interaction between *ReasonFlux* and $\pi_{inf}$ is not a one-way process but rather in an iterative manner. After obtaining the instantiated step $\hat{s_i}$, it is then evaluated and analyzed by *ReasonFlux*, and we represented this adjustment as process $\delta_i = \textit{ReasonFlux}(\mathbb{T}^*_{\text{traj}}, \hat{s_i})$. Based on this evaluated result and analysis, *ReasonFlux* decide whether to refine the trajectory, potentially adjusting subsequent steps or even retrieving alternative templates. This iterative refinement can be expressed as:

$$\mathbb{T}^*_{\text{traj}} \leftarrow \textit{ReasonFlux}(\mathbb{T}^*_{\text{traj}}, \delta_i). \tag{7}$$

This iterative feedback mechanism between *ReasonFlux* and $\pi_{inf}$ underscores a crucial aspect of complex problem-solving: the dynamic interplay between planning and execution. By analyzing intermediate results generated during the reasoning process, *ReasonFlux* gains valuable insights that can inform adjustments to the trajectory. This ability to refine the solution path precisely reflects how humans often uncover more efficient or effective solutions by examining partial results. Furthermore, intermediate steps may reveal previously obscured constraints or opportunities within the problem, allowing for a more informed and targeted approach. Therefore, the hierarchical nature of *ReasonFlux*, enabled by this iterative refinement, is crucial for navigating the complexities of challenging reasoning tasks and achieving optimal solutions. In summary, *ReasonFlux* achieves effective problem solving by dynamically configuring and adjusting the template trajectory based on the problem complexity, transcending the limitations of traditional inference methods and offering a more efficient and powerful reasoning framework.

## 4. Experiments

**Template Library Construction** As illustrated in Section 3.1, we use Gemini-2.0 (Team et al., 2023) to summarize and extracts high-level thoughts from the training sets of various math datasets, such as MATH (7.5K samples) (Lightman et al., 2023), and self-curated CN high-school competition-level data (2K samples), and construct our structured thought template library (approximately 500 thought templates). We provide some template examples in Appendix A.

**Training Details** Due to limited GPU resources, we use Qwen2.5-32B-Instruct (Yang et al., 2024a) as the base model and also adopt it as our inference LLM. In our training procedure, we **only use 8 NVIDIA A100 GPUs**, which is very cost-efficient. In the structure-based finetuning stage (Section 3.2), we train the initialized $\pi_{\text{struct}}$ with the training dataset $\mathcal{D}_{\text{train}}$ containing 15K samples extended from our template library $\mathcal{D}_{\text{temp}}$. We conduct the initialization training for 6 epochs using an AdamW optimizer along with the cosine learning rate scheduler. In the template trajectory optimization process (Section 3.2), we train our *ReasonFlux* with 10K collected pair-wise trajectories from MATH (7.5k), and self-curated CN high-school competition-level data (2K) for 6 epochs using an AdamW optimizer along with cosine learning rate scheduler.

**Evaluation Datasets** To evaluate the complex reasoning capabilities, we choose a broad set of challenging reasoning benchmarks, including MATH (Lightman et al., 2023), AIME 2024 (AI-MO, 2024a), AMC 2023 (AI-MO, 2024b), OlympiadBench (He et al., 2024) and GaoKao (Chinese College Entrance Exam) En 2023 (Liao et al., 2024). These benchmarks comprehensively evaluate mathematical reasoning capabilities, and they are all competition-level and Olympic-level problems. Moreover, AIME 2024 and AMC 2023 are highly challenging competition benchmarks, which are of limited sizes of test samples in AMC and AIME and the results are averaged over 16 runs.

*Table 2.* Pass@1 accuracy comparison on various mathematical reasoning benchmarks.

| Model | MATH | AIME 2024 | AMC 2023 | Olympiad Bench | Gaokao En 2023 |
|---|---|---|---|---|---|
| **Frontier LLMs** | | | | | |
| GPT-4o | 76.6 | 9.3 | 47.5 | 43.3 | 67.5 |
| Claude3.5-Sonnet | 78.3 | 16.0 | - | - | - |
| GPT-o1-preview | 85.5 | 44.6 | 90.0 | - | 71.4 |
| GPT-o1-mini | 90.0 | 56.7 | 95.0 | 65.3 | 78.4 |
| **Open-Sourced Reasoning LLMs** | | | | | |
| DeepSeek-Coder-V2-Instruct | 75.3 | 13.3 | 57.5 | 37.6 | 64.7 |
| Mathstral-7B-v0.1 | 57.8 | 0.0 | 37.5 | 21.5 | 46.0 |
| NuminaMath-72B-CoT | 64.0 | 3.3 | 70.0 | 32.6 | 58.4 |
| LLaMA3.1-8B-Instruct | 51.4 | 6.7 | 25.0 | 15.4 | 38.4 |
| LLaMA3.1-70B-Instruct | 65.4 | 23.3 | 50.0 | 27.7 | 54.0 |
| LLaMA3.1-405B-Instruct | 73.8 | - | - | 34.8 | - |
| Qwen2.5-Math-72B-Instruct | 85.6 | 30.0 | 70.0 | 49.0 | 71.9 |
| rStar-Math | 88.2 | 43.3 | 80.0 | 63.1 | 78.2 |
| DeepSeek-V3 | 90.2 | 39.2 | 80.0 | 55.4 | - |
| **ReasonFlux-32B** | **91.2** | **56.7** | **85.0** | **63.3** | **83.6** |
| *1.5B-Level Base Model* | | | | | |
| Qwen2.5-Math-1.5B | 51.2 | 0.0 | 22.5 | 16.7 | 46.5 |
| Qwen2.5-Math-1.5B-Instruct | 60.0 | 10.0 | 60.0 | 38.1 | 65.5 |
| **ReasonFlux-1.5B** | **70.4** | **20.0** | **72.5** | **49.0** | **76.6** |
| *7B-Level Base Model* | | | | | |
| Qwen2.5-Math-7B | 58.8 | 3.3 | 22.5 | 21.8 | 51.7 |
| SuperCorrect-7B | 70.2 | 10.0 | 37.5 | 39.0 | 64.0 |
| Qwen2.5-Math-7B-Instruct | 82.6 | 13.3 | 62.5 | 41.6 | 66.8 |
| **ReasonFlux-7B** | **88.6** | **36.7** | **80.0** | **54.8** | **80.5** |
| *32B-Level Base Model* | | | | | |
| Qwen2.5-32B-Instruct | 79.4 | 16.5 | 64.0 | 45.3 | 72.1 |
| QwQ-32B-preview | 90.6 | 50.0 | 75.0 | - | 65.3 |
| Sky-T1-32B-preview | 86.4 | 43.3 | - | 59.8 | - |
| **ReasonFlux-32B** | **91.2** | **56.7** | **85.0** | **63.3** | **83.6** |

**Baselines** To demonstrate reasoning ability of *ReasonFlux*, we compare it with two kinds of strong baseline models: **(i)** *Frontier LLMs* contain GPT-4o, Claude, OpenAI o1-preview and o1-mini. We report their performance on our evaluation benchmarks by taking accuracy numbers from different public technical reports. **(ii)** *Open-sourced superior reasoning models* contain DeepSeek-Coder-v2-Instruct, Mathstral (Team, 2024b), NuminaMath-72B (Li et al., 2024), LLaMA3.1 (Dubey et al., 2024), Qwen2.5-Math (Yang et al., 2024a), SuperCorrect-7B-Instruct (Yang et al., 2024c), QwQ-32B-Preview (Team, 2024a), rStar-Math (Guan et al., 2025) and Sky-T1-32B-Preview (distilled from QwQ-32B-Preview), and DeepSeek-V3 (Liu et al., 2024a), which are widely used and followed open-sourced reasoning models. Both kinds of baselines represent the highest level of mathematical reasoning currently available.

### 4.1. Results on Challenging Reasoning Benchmarks

Table 2 shows the final results of our *ReasonFlux* with a comprehensive comparison to SOTA reasoning models. We find that our *ReasonFlux*-32B consistently outperforms both frontier LLMs and open-sourced reasoning LLMs on most challenging mathematical benchmarks, achieving new SOTA performances with only 32B-level parameters. More specifically, on the MATH benchmark, *ReasonFlux* achieves 91.2% of accuracy, **surpassing frontier reasoning models o1-preview by 6.7%**, and current SOTA-level open-source LLMs **with only 32B parameters**. On the AIME 2024 benchmark, *ReasonFlux* consistently demonstrates its extrodinary reasoning capabilities with 56.7% accuracy, **significantly surpassing o1-preview and DeepSeek-V3 by 27% and 45%**, respectively, and matching the performance of the proprietary OpenAI o1-mini. On the AMC 2023 benchmark, our method, *ReasonFlux*, maintains its position within the top tier of all reasoning LLMs with 85.0% accuracy, significantly outperforming other open-source LLMs while achieving performance comparable to

proprietary LLMs. This further validates the effectiveness of our approach in mathematical reasoning and underscores its substantial potential for further development and application. We provide some reasoning details in Section 4.3.

Beyond above well-known benchmarks, *ReasonFlux*-32B also demonstrates impressive generalization and effectiveness on other challenging datasets. Notably, it achieves a 63.3% accuracy **on OlympiadBench surpassing DeepSeek-V3 by 14%**, and an 83.6% accuracy **on the Chinese College Entrance Mathematics Exam (Gaokao) surpassing o1-mini by 7%**. These results are particularly noteworthy because our template library was constructed primarily from publicly available datasets, the same template library was used consistently across all evaluation processes. This consistent strong performance across diverse and challenging mathematical reasoning tasks, ranging from competition-level problems to standardized exams, provides compelling evidence for the robust generalization ability and effectiveness of *ReasonFlux*. It underscores the power of our template-driven approach to capture and apply underlying mathematical principles, regardless of the specific format or context of the problem.

**Generalizing to Different Base Models** From Table 2, we also observe that our *ReasonFlux* can achieve consistent and significant improvement across all evaluation benchmarks when using different base models as both navigator and inference LLM. Notably, our *ReasonFlux* usually achieves even surpasses the reasoning accuracy of the models in next level. These phenomenons demonstrate both effectiveness and generalization ability of our *ReasonFlux*.

*Table 3.* Generalization ability of our thought templates with different base LLMs on a series of similar mathematical problems.

| Model | direct reasoning (%) | with Template (%) |
|---|---|---|
| **Llama-3.1-8B-Instruct** | 47.6 | 75.1 (**+27.5**) |
| **Qwen2.5-7B-Instruct** | 59.2 | 82.7 (**+23.5**) |
| **Qwen2.5-Math-7B-Instruct** | 66.5 | 88.4 (**+21.9**) |
| **Llama-3.1-70B-Instruct** | 67.4 | 91.2 (**+23.8**) |
| **Qwen2.5-32B-Instruct** | 69.2 | 94.3 (**+25.1**) |
| **Qwen2.5-Math-32B-Instruct** | 71.1 | 95.9 (**+24.8**) |

### 4.2. Generalization Ability of Structured Template Library

We presents additional experiments on MATH benchmark designed to evaluate the generalization ability of our structured template library. To achieve this, we randomly sampled 100 templates from the library, each paired with its corresponding example problem. Subsequently, we employed o1-preview to generate 50 variant problems for each example. These variants were carefully constructed to ensure they differed from the original examples while still assessing the same underlying knowledge and skills.

We then used these templates as in-context examples to guide different LLMs during inference on the generated variant problems. We compare the average accuracy between our template augmented reasoning and direct reasoning (i.e., solving the problems without template). As illustrated in Table 3, our template-augmented approach significantly improves the reasoning accuracy of different base models compared to direct reasoning. This demonstrates the ability of our structured templates to generalize effectively across a range of similar problems, rather than being limited to specific instances. Furthermore, we observed that smaller-sized LLMs, when guided by our templates, were able to outperform larger-sized LLMs employing direct reasoning. This finding underscores the effectiveness and high quality of our structured template library.

### 4.3. Reasoning Flows over Planned Template Trajectory

We showcase detailed examples of our reasoning flows, as depicted in Figure 3, when tackling challenging mathematical problems. Specifically, ReasonFlux begins by meticulously observing and analyzing the input problem, engaging in deep thought to explore potential solution pathways. Based on this initial assessment, ReasonFlux intelligently configures a dynamic reasoning trajectory, strategically retrieving relevant templates from our structured template library to guide each logical step. Then, ReasonFlux initiates an interactive instruction with the inference LLM, guiding it to follow the prescribed trajectory and execute the reasoning process along the trajectory. Crucially, the results obtained from preceding steps are seamlessly integrated as contextual information, informing and conditioning the subsequent steps. Compare to conventional self-explore and reasoning paradigm, our method could consistently improve the reasoning accuracy and efficiency.

## Problem:

Let $x$, $y$, and $z$ be positive real numbers satisfying the system of equations:
$\sqrt{2x-xy} + \sqrt{2y-xy} = 1$, $\sqrt{2y-yz} + \sqrt{2z-yz} = \sqrt{2}$, $\sqrt{2z-zx} + \sqrt{2x-zx} = \sqrt{3}$ .
Then $[(1 - x)(1 - y)(1 - z)]^2$ can be written as $\frac{m}{n}$, where $m$ and $n$ are relatively prime positive integers. Find $m + n$.

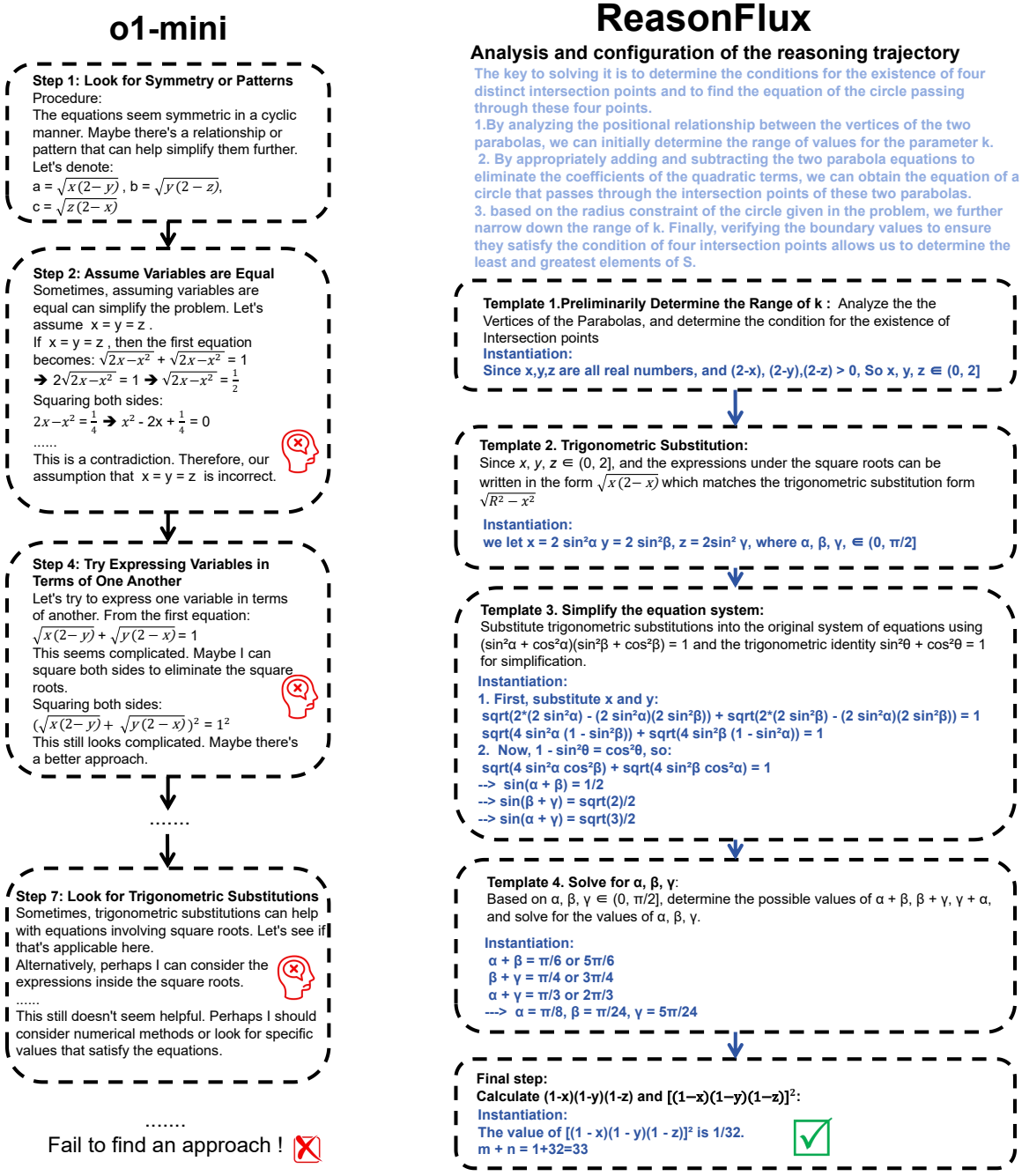


*Figure 3.* Comprasion between o1-mini and *ReasonFlux*.

### 4.4. Inference Scaling Laws for Template-Augmented Reasoning

Different from traditional inference scaling with Best-of-$N$ and Majority Voting (Wu et al., 2024), our *ReasonFlux* owns a specific interplay-based scaling mechanism. In order to provide a comprehensive understanding of how *ReasonFlux* automatically trade off between cost and performance. As shown in Figure 4, we demonstrate (i) how number of retrieved templates adaptively scales with increased problem complexity and (ii) how rounds of interplay between *ReasonFlux* and inference LLMs adaptively scales with increased problem complexity. From the results, we can observe that our *ReasonFlux* can effectively capture the complexity of input problems, and plan out reasonable template trajectories with appropriate interplay rounds. Utilizing more fine-grained thought templates may boost the scaling effect of our *ReasonFlux*, and we leave this exploration for future work.

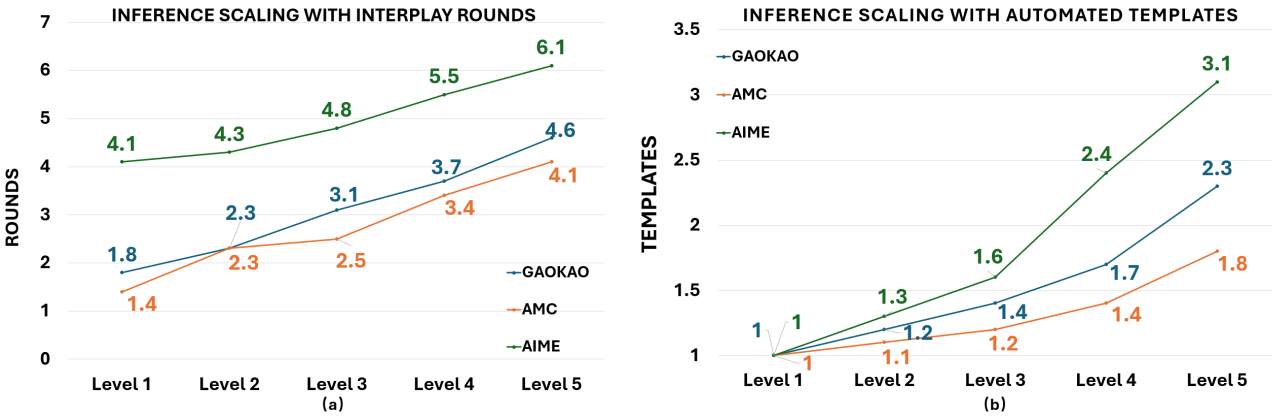


*Figure 4.* **Inference scaling laws for template-augmented reasoning in *ReasonFlux*.** (a) Scaling interplay rounds between planning and instantiation with increased level of problem complexity. (b) Scaling retrieved templates with increased level of problem complexity.

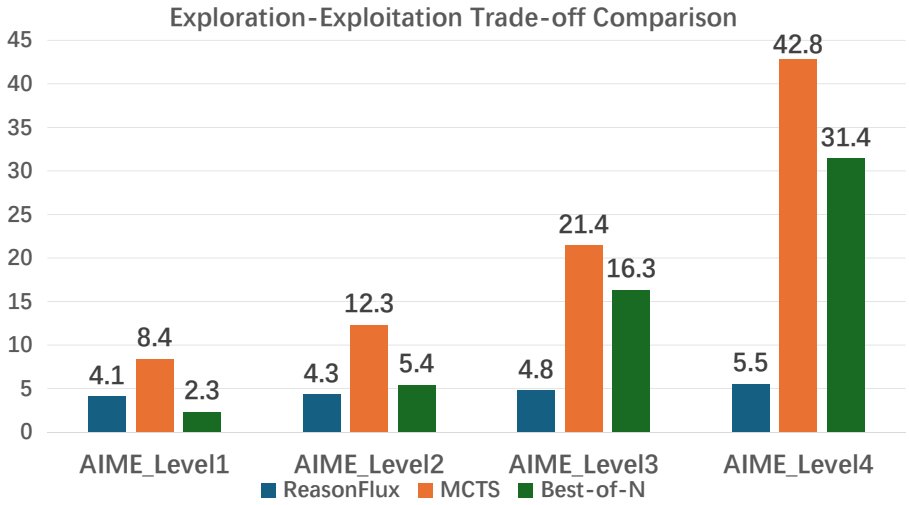


*Figure 5.* **Exploration-Exploitation Trade-off Comparison between different reasoning strategies.** Here we experiment with a diverse set of 200 problems sourced from the AIME competitions spanning 1983 to 2023, divided into four difficulty levels. We test the average exploration cost of ReasonFlux (number of interplay rounds), MCTS (number of reasoning steps) and Best-of-N (number of reasoning trajectories).

### 4.5. Better Exploration-Exploitation Trade-off

To evaluate the exploration-exploitation trade-off of different reasoning strategies, we conducted an ablation study comparing our proposed interplay method against Best-of-N and MCTS. Each method exhibits a distinct approach to navigating the reasoning space. Best-of-N constructs multiple reasoning trajectories to identify the optimal path, while MCTS iteratively explores the most promising next step during the problem-solving process.

Our method formulates a potential reasoning trajectory and then guides the interactive process with the inference LLM for iterative refinement and adjustments. To ensure a fair comparison, we introduce a unified metric termed "exploration-exploitation cost." This metric quantifies the number of exploration attempts required by each method to correctly solve a given problem. For our method, this denotes the number of interactions between ReasonFlux and the inference LLM. For MCTS, it is represented by the iteration time, and for Best-of-N, it denotes the total number of sampled trajectories.

As illustrated in Figure 5, both MCTS and Best-of-N exhibit an increasing exploration-exploitation cost as problem difficulty escalates. In contrast, our method maintains a consistently lower and more stable exploration cost across all difficulty levels. This superior efficiency of our method can be attributed to the effectiveness of our structured template library. This high-quality library effectively refines the search space, facilitating the identification of correct reasoning paths. Furthermore, the high quality and generalization ability of the templates (experimental analysis in Section 4.2) within the library allows for effective exploitation, guiding the Inference LLM towards accurate and efficient reasoning. Consequently, our approach demonstrates a more balanced and efficient exploration-exploitation trade-off compared to Best-of-N and MCTS.

## 5. Conclusion

In this work, we present *ReasonFlux*, a new hierarchical LLM reasoning framework that adaptively scales fundamental and essential thought templates for simplifying the search space of complex reasoning, and outperforming the mathematical reasoning capabilities of powerful LLMs like OpenAI o1-preview and DeepSeek V3. We introduces a structured and compact thought template library, hierarchical reinforcement learning on thought template trajectory and a brand new inference scaling system. Extensive experiments across different challenging math benchmarks demonstrate the superiority of *ReasonFlux*. We also reveal some key findings, including the scaling laws for our template-augmented reasoning and the superior exploration-exploitation trade-off of our *ReasonFlux* over previous reasoning strategies.

## References

AI-MO. Aime 2024, 2024a. URL https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AI-MO. Amc 2023, 2024b. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Asai, A., Min, S., Zhong, Z., and Chen, D. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pp. 41–46, 2023.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Choi, S., Fang, T., Wang, Z., and Song, Y. Kcts: knowledge-constrained tree search decoding with token-level hallucination detection. *arXiv preprint arXiv:2310.09044*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Guan, X., Zhang, L. L., Liu, Y., Shang, N., Sun, Y., Zhu, Y., Yang, F., and Yang, M. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z. L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024.

Hong, J., Lee, N., and Thorne, J. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, 2024.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Lai, X., Tian, Z., Chen, Y., Yang, S., Peng, X., and Jia, J. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, J., Beeching, E., Tunstall, L., Lipkin, B., Soletskyi, R., Huang, S., Rasul, K., Yu, L., Jiang, A. Q., Shen, Z., et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13, 2024.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, 2023.

Liao, M., Luo, W., Li, C., Wu, J., and Fan, K. Mario: Math reasoning with code interpreter output–a reproducible pipeline. *arXiv preprint arXiv:2401.08190*, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

Liu, J., Cohen, A., Pasunuru, R., Choi, Y., Hajishirzi, H., and Celikyilmaz, A. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. In *First Conference on Language Modeling*, 2024c.

Luo, L., Liu, Y., Liu, R., Phatale, S., Lara, H., Li, Y., Shu, L., Zhu, Y., Meng, L., Sun, J., et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

Mialon, G., Dessi, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Roziere, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Qi, Z., Ma, M., Xu, J., Zhang, L. L., Yang, F., and Yang, M. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195*, 2024.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., and Yih, W.-t. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*, 2023.

Shi, W., Yuan, M., Wu, J., Wang, Q., and Feng, F. Direct multi-turn preference optimization for language agents. *arXiv preprint arXiv:2406.14868*, 2024.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Swamy, G., Dann, C., Kidambi, R., Wu, Z. S., and Agarwal, A. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024.

Tang, H., Hu, K., Zhou, J. P., Zhong, S., Zheng, W.-L., Si, X., and Ellis, K. Code repair with llms gives an exploration-exploitation tradeoff. *arXiv preprint arXiv:2405.17503*, 2024.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: A family of highly capable multimodal models, 2024.

Team, Q. Qwq: Reflect deeply on the boundaries of the unknown, November 2024a. URL https://qwenlm.github.io/blog/qwq-32b-preview/.

Team, T. M. A. Mathstral-7b-v0.1, 2024b. URL https://huggingface.co/mistralai/Mathstral-7B-v0.1.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*, 2023.

Wang, Z., Nie, W., Qiao, Z., Xiao, C., Baraniuk, R., and Anandkumar, A. Retrieval-based controllable molecule generation. In *The Eleventh International Conference on Learning Representations*, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.

Yang, L., Huang, Z., Zhou, X., Xu, M., Zhang, W., Wang, Y., Zheng, X., Yang, W., Dror, R. O., Hong, S., et al. Prompt-based 3d molecular diffusion models for structure-based drug design. 2023.

Yang, L., Yu, Z., Zhang, T., Cao, S., Xu, M., Zhang, W., Gonzalez, J. E., and Cui, B. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 2024b.

Yang, L., Yu, Z., Zhang, T., Xu, M., Gonzalez, J. E., Cui, B., and Yan, S. Supercorrect: Supervising and correcting language models with error-driven insights. *arXiv preprint arXiv:2410.09008*, 2024c.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yasunaga, M., Aghajanyan, A., Shi, W., James, R., Leskovec, J., Liang, P., Lewis, M., Zettlemoyer, L., and Yih, W.-T. Retrieval-augmented multimodal language modeling. In *International Conference on Machine Learning*, pp. 39755–39769. PMLR, 2023.

Zhang, D., Zhoubian, S., Hu, Z., Yue, Y., Dong, Y., and Tang, J. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*, 2024a.

Zhang, S., Chen, Z., Shen, Y., Ding, M., Tenenbaum, J. B., and Gan, C. Planning with large language models for code generation. *International Conference on Machine Learning*, 2024b.

Zhang, T., Patil, S. G., Jain, N., Shen, S., Zaharia, M., Stoica, I., and Gonzalez, J. E. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*, 2024c.

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., and Cui, B. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

## A. More Examples of Structured Template Library

In this section, we present a more detailed and diverse collection of supplementary examples for Section 3.1, showcasing our meticulously designed structured templates. These examples span a range of template types, demonstrating the versatility and applicability of our approach. The template types include: 1) **Problem-Solving Methods**, which provide step-by-step procedures for tackling specific problem types; 2) **Secondary Mathematical Conclusions**, which encapsulate derived mathematical results that can be applied to various problems; 3) **Property & Theorem** that highlight essential mathematical properties and theorems; 4) **Knowledge Application** templates that demonstrate the application of specific mathematical concepts and techniques; and 5) **Important Formulas and Rules** templates, which offer concise summaries of crucial formulas and rules for quick reference and application.

To emphasize the structure and facilitate comprehension, each template is designed to contain two kinds of data: i) **Template Metadata** and ii) **Template Content**. The Template Metadata provides concise information about the template, including its **name** $T_{\text{nam}}$**, relevant knowledge tags** $T_{\text{tag}}$**, a brief description** $T_{\text{des}}$**, and typical application scenarios** $T_{\text{sco}}$. This section serves as a quick reference guide, enabling LLMs to efficiently locate and identify templates relevant to their needs. The Template Content delves into the core of the template, presenting the detailed reasoning flow and a concrete example illustrating its application. **The reasoning flow corresponding to the application steps** $T_a$ **and the example application corresponding to** $T_{\text{exa}}$ **in Section 3.1**, which outlines the logical steps or procedures involved in utilizing the template, while the example provides a practical demonstration of how the template can be applied to solve a specific problem. This two-part structure enhances clarity and allows for both quick retrieval and in-depth understanding of each template.

The following examples have been carefully selected to provide a comprehensive overview of the capabilities of our structured template library. Through these examples, we aim to more comprehensive overview of our structured templates, and demonstrate the effectiveness of our structured templates in promoting organized thinking, facilitating problem-solving, and ultimately enhancing mathematical understanding of LLMs.

(I) Template ( Problem-Solving Method) : Five-Step Method for Solving Absolute Value Inequalities

**Template Name:** Five-Step Method for Solving Absolute Value Inequalities
**Knowledge Tag:** Absolute Value Inequalities, Solving Inequalities, Combining Numerical and Graphical Methods
**Description:** This template provides a structured approach to solving absolute value inequalities using various strategies, with a focus on the squaring method and the zero-point interval method.
**Application Scenario:** Applicable to absolute value inequalities of the form $|x-a|>b$, $|ax+b|<c$, $|f(x)|>|g(x)|$, etc. Particularly suitable for complex cases involving multiple absolute value symbols or requiring interval discussions.
**Reasoning Flow:**

1. Standardize the inequality to ensure the right side is non-negative (e.g., $|x-1|>|2x+3|$).
2. Choose a solution strategy (Step 3 will present two options).
3. **Solve using one of the following methods:**
   (a) **Squaring Method:**
      (i) Rearrange to the form $A^2>B^2$.
      (ii) Expand and simplify into a polynomial inequality.
      (iii) Factor, find the roots, and use a number line to determine the solution set.
   (b) **Interval Method:**
      (i) Mark the zero points of each absolute value expression (e.g., $x=1$ and $x=-1.5$).
      (ii) Divide the number line into intervals (e.g., $x\leq -1.5$, $-1.5<x<1$, $x\geq 1$).
      (iii) Rewrite the inequality without absolute value signs within each interval.
      (iv) Solve the inequality in each interval and find the intersection with the interval.
4. Verify whether the endpoint values satisfy the original inequality.
5. Combine the solution sets from each interval, expressing the final result using set notation. (If using the interval method)

**Example Application:**
**Problem:** Solve the inequality $|x-1|>|2x+3|$.
**Solution Process:**
Using Squaring Method (Step 3a):

1. Square both sides: $(x-1)^2>(2x+3)^2$
2. Expand and simplify: $x^2-2x+1>4x^2+12x+9\rightarrow -3x^2-14x-8>0$
3. Factor: $-(3x+2)(x+4)>0\rightarrow (3x+2)(x+4)<0$
4. Find the roots and use a number line: $x=-4$, $x=-\frac{2}{3}\rightarrow$ Solution set: $(-4,-\frac{2}{3})$

Using Interval Method (Step 3b):
To better present our templates, we have omitted some examples that were too long. .....

(II) Template (Secondary Conclusion) : Application of the Inequality of Arithmetic and Geometric Means for Three and n Variables

**Template Name:** Application of the Inequality of Arithmetic and Geometric Means for Three and n Variables
**Knowledge Tag:** Inequality of Arithmetic and Geometric Means, Three-Variable Inequality, n-Variable Inequality, Inequality Proof
**Description:** Extends the two-variable inequality of arithmetic and geometric means to three and n variables, suitable for handling the relationship between the sum and product of multiple positive numbers. The core formulas are: for three variables, $a^3 + b^3 + c^3 \geq 3abc$; for $n$ variables, the arithmetic mean is greater than or equal to the geometric mean.
**Application Scenario:** Used when there are three or more positive variables in the problem, and it is necessary to compare the relationships between sum, product, sum of squares, etc. Especially suitable for proving inequalities with multiple variables or finding the maximum/minimum values.
**Reasoning Flow:**

1. Confirm that all variables are positive (ensure this through the problem's conditions or transformations if necessary).
2. If it is a three-variable case, directly apply $a^3 + b^3 + c^3 \geq 3abc$ (equality holds if and only if $a = b = c$).
3. If it is an $n$-variable case, apply the inequality of arithmetic and geometric means:

$$\frac{a_1 + a_2 + ... + a_n}{n} \geq \sqrt[n]{a_1 a_2 ... a_n}$$

(equality holds if and only if $a_1 = a_2 = ... = a_n$).
4. Transform the original expression into the standard form above through algebraic manipulations (such as grouping, factoring, completing the square, etc.).
5. Combine with known conditions (such as $abc = 1$) to substitute and simplify to find the maximum/minimum value.
6. Verify that the condition for equality holds satisfies the problem's constraints.

**Example Application:**
**Problem:** Given that $a$, $b$, and $c$ are positive numbers and $abc = 1$, prove that $(a + b)^3 + (b + c)^3 + (c + a)^3 \geq 24$.
**Solution:**

1. Confirm $a, b, c > 0$ and $abc = 1$.
2. Apply the three-variable inequality to each term in parentheses: $(a + b)^3 \geq 8ab(a + b)/8$ (needs to be adjusted to fit the form).
3. Better solution: Directly apply $a^3 + b^3 + c^3 \geq 3abc$.

$$\because (a + b)^3 + (b + c)^3 + (c + a)^3 \geq 3(a + b)(b + c)(c + a)$$

4. Apply the two-variable inequality of arithmetic and geometric means to $(a + b)(b + c)(c + a)$:

$$(a + b) \geq 2\sqrt{ab}, (b + c) \geq 2\sqrt{bc}, (c + a) \geq 2\sqrt{ca}$$

$$\therefore \text{The product} \geq 8\sqrt{a^2b^2c^2} = 8abc = 8$$

5. Substitute to get the original expression $\geq 3 \times 8 = 24$.
6. Verify the equality condition: Equality holds if and only if $a = b = c = 1$.

(III) Template (Property Theorem) : Extremum Value Theorem

**Template Name:** Extremum Value Theorem
**Knowledge Tag:** Inequality of Arithmetic and Geometric Means, Extremum Value Theorem, Product is Maximum when Sum is Constant, Sum is Minimum when Product is Constant
**Description:** When the product or sum of two positive numbers $x$ and $y$ is a constant, their sum or product has an extremum value: when the product is constant, the sum has a minimum value; when the sum is constant, the product has a maximum value. Equality holds if and only if $x = y$.
**Application Scenario:** Suitable for finding the maximum/minimum value of the sum or product of two positive variables, especially when the product or sum of one of the expressions is a constant. For example: rectangle perimeter/area problems, function optimization problems, etc.
**Reasoning Flow:**

1. Confirm that variables $x$ and $y$ are both positive.
2. Determine if there is a constant product $xy = P$ or a constant sum $x + y = S$ in the problem.
3. If the product is a constant $P$, then the minimum value of the sum $x + y$ is $2\sqrt{P}$ (when and only when $x = y$).
4. If the sum is a constant $S$, then the maximum value of the product $xy$ is $\frac{S^2}{4}$ (when and only when $x = y$).
5. Verify that the condition for equality holds satisfies the problem's requirements (e.g., the actual range of values for $x$ and $y$).

**Example Application:**
**Problem:** What is the minimum value of the function $y = \frac{x^4-5x^2+1}{x^2-5}$ $(x^2 > 5)$?
**Solution:**

1. Confirm the variable is positive: $x^2 > 5 \Rightarrow x^2 - 5 > 0$.
2. Transform the function: $y = x^2 + \frac{1}{x^2-5} - 5$.
3. Let $a = x^2 - 5 > 0$, then $y = a + \frac{1}{a}$.
4. Apply the Extremum Value Theorem: $a + \frac{1}{a} \geq 2\sqrt{a \cdot \frac{1}{a}} = 2$ (when and only when $a = \frac{1}{a} \Rightarrow a = 1$).
5. Therefore $y \geq 2 + 5 = 7$, when and only when $x^2 - 5 = 1 \Rightarrow x = \pm\sqrt{6}$, the equality holds.

**Answer:** 7

(IV) Template (Knowledge Application) : Analyzing the Parity and Symmetry of Trigonometric Functions Using Reduction Formulas

**Template Name:** Analyzing the Parity and Symmetry of Trigonometric Functions Using Reduction Formulas
**Knowledge Tag:** Reduction Formulas, Parity, Symmetry, Properties of Trigonometric Functions
**Description:** This template guides the analysis of the parity and symmetry of complex trigonometric functions by transforming them into standard forms using reduction formulas, aiding students in systematically solving related problems.
**Application Scenario:** Applicable to determining the parity of trigonometric functions, identifying the symmetry centers or axes of function graphs, and solving for parameters (e.g., phase angle $\phi$). This method is useful when encountering functions of the form $y = A\sin(\omega x + \phi)$ or $y = A\cos(\omega x + \phi)$.
**Reasoning Flow:**

1. Transform the target trigonometric function into a standard sine or cosine form using reduction formulas. For example, use $\sin(x + \frac{\pi}{2}) = \cos x$ to convert a cosine function to a sine form.
2. Determine the function's parity based on the definition of odd and even functions. An odd function satisfies $f(-x) = -f(x)$, and an even function satisfies $f(-x) = f(x)$.
3. If symmetry is involved, determine the expressions for the symmetry axes or centers. For example, the symmetry axes of the sine function are $x = \frac{\pi}{2} + k\pi$, and the symmetry centers are $(k\pi, 0)$.
4. Compare the transformed function with the standard form and solve the equation to find the unknown parameters (e.g., $\phi$). For example, set the phase angle to satisfy the condition for an odd function, $\phi = k\pi$.
5. Verify the solution's validity, ensuring it conforms to the original function's domain and fundamental properties.

**Example Application:**
**Problem:** Given that the function $y = \sqrt{2}\sin(x + \phi)$ is an odd function, find the possible values of $\phi$.
**Solution Steps:**

1. Based on the definition of an odd function, we have $\sqrt{2}\sin(-x + \phi) = -\sqrt{2}\sin(x + \phi)$.
2. Expand the left side: $\sin(-x + \phi) = \sin\phi\cos x - \cos\phi\sin x$.
3. Simplify the right side: $-\sin(x + \phi) = -\sin x\cos\phi - \cos x\sin\phi$.
4. Compare the coefficients on both sides of the equation: $\sin\phi = -\sin\phi$ and $-\cos\phi = -\cos\phi$.
5. Solve for $\phi$: $\sin\phi = 0 \Rightarrow \phi = k\pi$ ($k \in \mathbb{Z}$).

(V) Template (Important Formulas/Rules) : Distance Formulas and Their Applications

**Template Name:** Distance Formulas and Their Applications
**Knowledge Tag:** Distance Between Two Points, Distance from a Point to a Line, Distance Between Parallel Lines
**Description:** This template includes formulas for calculating three types of distances: the distance between two points, the distance from a point to a line, and the distance between two parallel lines. These formulas are core tools for solving distance problems in analytic geometry.
**Application Scenario:** This template can be applied when it is necessary to calculate the geometric distance between two points, the perpendicular distance from a point to a line, or the fixed distance between two parallel lines. It is commonly used in scenarios such as calculating the area of geometric figures, analyzing positional relationships, and solving symmetry problems.
**Reasoning Flow:**

1. Step 1: Identify the type of problem (distance between two points / distance from a point to a line / distance between parallel lines).
2. Step 2: Distance between two points formula: $|P_1P_2| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, substitute the coordinates directly to calculate.
3. Step 3: Distance from a point to a line formula: $d = \frac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}}$, ensure the line equation is in the general form $Ax + By + C = 0$.
4. Step 4: Distance between parallel lines formula: $d = \frac{|C_1-C_2|}{\sqrt{A^2+B^2}}$, both line equations must be in the form $Ax + By + C_1 = 0$ and $Ax + By + C_2 = 0$ with the same coefficients $A$ and $B$.
5. Step 5: Handle special cases (e.g., projection distance on coordinate axes, distance transformation in symmetry problems).

**Example Application:**
**Problem:** Given that the line $l_1 : mx + 2y - 4 - m = 0$ has equal intercepts on the x-axis and y-axis, find the distance between $l_1$ and $l_2 : 3x + 3y - 1 = 0$.
**Solution Steps:**

1. From equal intercepts, we get $\frac{m+4}{m} = \frac{m+4}{2} \Rightarrow m = 2$.
2. Convert $l_1$ to the general form $2x + 2y - 6 = 0 \Rightarrow x + y - 3 = 0$.
3. Align coefficients: Rewrite $l_1$ as $3x + 3y - 9 = 0$ to match the coefficients of $l_2$.
4. Apply the parallel lines distance formula $d = \frac{|-1-(-9)|}{\sqrt{3^2+3^2}} = \frac{8}{3\sqrt{2}} = \frac{4\sqrt{2}}{3}$.

**Answer:** $\frac{4\sqrt{2}}{3}$