FLAIR: Steering LLM Mathematical Problem Solving based on A Fuzzy-Logic-AssIsted Reasoner

Hao Wu[1,2,*], Hongru Sun[2,*], Wanqing Li[2], Xinguo Yu[1,†], Hao Ming[1], Xiao Luo[3], Wenbin Zhang[4], Jiahong Zhao [5], Yi Guo[6,†], Jie Yang[2,†]

[1]Central China Normal University, China [2]University of Wollongong, Australia
[3]University of Wisconsin-Madison, USA [4]Florida International University, USA
[5]University of Southampton, UK [6]Western Sydney University, Australia

Correspondence:hwu@mails.ccnu.edu.cn, jiey@uow.edu.au *Equal Contribution †Corresponding Author

Abstract

Mathematical reasoning is one of the core capabilities for Large Language Models (LLMs). Yet, existing approaches often rely on static heuristics or pre-determined reasoning strategies, limiting their ability to adapt to different intermediate states. To address this limitation, we propose **FLAIR** (**F**uzzy-**L**ogic-**A**ss**I**sted **R**easoner), an adaptive framework that integrates fuzzy theory into LLM-based mathematical reasoning. Specifically, FLAIR characterizes intermediate problem-solving states using fuzzy memberships and employs a parameterized fuzzy rule system to conditionally activate subsequent actions. These rule parameters are further adjusted via Reinforcement Learning using solution-level feedback as the reward signal, enabling adaptive and iterative refinement without reliance on a fixed strategy. To the best of our knowledge, this work is the first to integrate fuzzy theory into LLM-based mathematical reasoning. Extensive experiments across multiple benchmarks demonstrate that FLAIR consistently outperforms recent state-of-the-art baselines, while offering effective and interpretable diagnostics of intermediate problem-solving states.

1 Introduction

The advent of Large Language Models (LLMs) has significantly reshaped the landscape of Natural Language Processing (Ouyang et al., 2022; Team, 2025). Among their emerging capabilities, *mathematical reasoning* is widely regarded as a core indicator of advanced logical competence (Yan et al., 2025), motivating substantial research efforts to investigate and improve the mathematical reasoning ability of LLMs.

Existing approaches differ in how and when they *regulate* the reasoning process, ranging from inference-time prompting and post-hoc verification to training-time model adaptation and tool-augmented execution. They can be broadly categorized into *Prompt Engineering* (Huang et al., 2024; Yin et al., 2025), *Reframing* (Zheng et al., 2024; Zhou et al., 2024), *Reasoning Strategies* (Yang et al., 2025; Leang et al., 2025; Cao et al., 2025), *Model Fine-tuning* (Liu et al., 2025c; Pei et al., 2025), *External Tool Integration* (J. Liu et al., 2025; Yao and Yadav, 2025), and *Verification* (Wu et al., 2025; Jiang et al., 2024a).

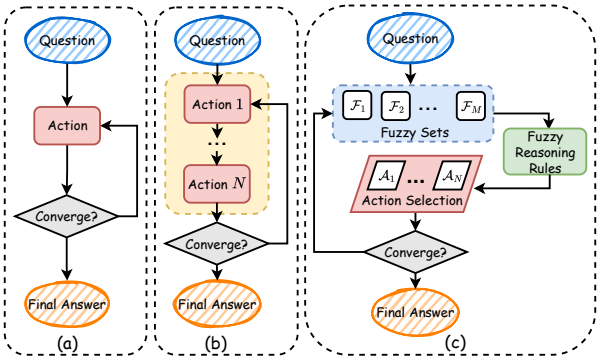


Figure 1: Illustration of different reasoning approaches: (a) single-paradigm reasoning, (b) pre-determined execution pipeline, and (c) the proposed adaptive fuzzy-enhanced framework.

Despite strong empirical performance, existing LLM-based methods struggle to effectively handle the inherent *uncertainty* in problem solving. That is, multiple reasoning deficiencies (*e.g.*, partial inconsistency, missing problem statements, or incorrect calculations) may co-exist within a single reasoning trajectory, each exhibiting different degrees of severity. However, most existing approaches adopt either a single-paradigm reasoning strategy (as shown in Fig. 1(a)) or a pre-determined execution pipeline (as shown in Fig. 1(b)), regardless of the underlying problem-solving uncertainty. Such *one-size-fits-all* designs lack the ability to explicitly diagnose intermediate reasoning states, which fundamentally restricts fine-grained and adaptive control under varying levels of reasoning difficulty and heterogeneous error patterns across tasks.

Addressing this limitation requires a principled

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 38638–38661
July 2-7, 2026

framework capable of modeling overlapping reasoning states and supporting uncertainty-aware adaptive control. **Fuzzy theory** (Zadeh, 1965) naturally meets this need by representing system states with graded membership values and enabling interpretable decision-making through fuzzy rules.

Accordingly, this paper introduces the **F**uzzy-**L**ogic-**A**ss**I**sted **R**easoner (FLAIR). Unlike existing approaches, FLAIR forms a dynamic feedback loop (Fig. 1(c)) that adaptively controls the reasoning process. Specifically, intermediate problem-solving states are first characterized by fuzzy membership estimation, after which parameterized Fuzzy Reasoning Rules (FRRs) conditionally activate appropriate actions. The rule parameters are further optimized via Reinforcement Learning (RL) based on solution-level feedback, enabling effective state–action policy learning and iterative refinement. In summary, the main contributions of this work are as follows:

- We propose FLAIR, which, to the best of our knowledge, is the first framework to integrate **fuzzy theory** with **LLM-based mathematical reasoning**, providing a principled approach to modeling and handling uncertainty during LLM reasoning.
- We design a fuzzy inference mechanism that formalizes reasoning-state diagnosis and correction via fuzzy sets and reasoning rules, enabling adaptive action selection beyond static heuristic triggering.
- We introduce a RL–based optimization strategy to automatically refine rule parameters, allowing the system to progressively learn effective state–action policies.
- We conduct extensive experiments on five widely-used benchmarks with six base LLMs, demonstrating that FLAIR consistently outperforms recent state-of-the-art methods in mathematical reasoning tasks.

2 Related Work

Mathematical Reasoning with LLMs. With the rapid development of LLMs, models such as ChatGPT (Ouyang et al., 2022) and Gemini (Team, 2025) demonstrate strong capabilities in a wide range of language tasks, particularly in mathematical reasoning. Existing studies on LLM-based mathematical reasoning generally fall into six categories: Prompt Engineering, Reframing, Reasoning Strategies, Model Fine-tuning, External Tool Integration, and Verification, as illustrated in Fig. 2[1].

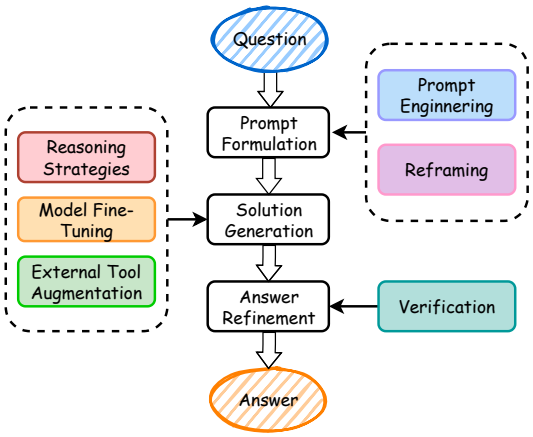


Figure 2: Conceptual map for LLM-based mathematical reasoning.

(1) Prompt Engineering methods elicit mathematical reasoning by designing and injecting exemplary reasoning chains into the prompt. Chain-of-Thought prompting (Diao et al., 2024) provides explicit step-by-step exemplars to guide deduction, while subsequent work simplifies demonstrations to retain the most salient examples (Huang et al., 2024) or selects error-matched exemplars to align with the target problem (Yin et al., 2025).

(2) Reframing techniques re-interpret the original question to improve semantic clarity and reduce ambiguity prior to reasoning (Zheng et al., 2024). Representative approaches leverage dependency graphs and entity linking (Liang et al., 2023), or generate paraphrased variants to examine solution consistency (Raiyan et al., 2023; Zhou et al., 2024).

(3) Reasoning Strategies explicitly structure the problem-solving process into verifiable steps. Representative methods are proposed to operate at the sub-expression level to construct fine-grained reasoning chains (Zhang et al., 2024a), formulate reasoning as a Markov process (Yang et al., 2025), or enforce symbolic conversion at each step to ensure algebraic correctness (Leang et al., 2025).

(4) Model Fine-tuning adapts LLMs through supervised or Reinforcement Learning–based training. Some methods introduce structured reward signals (Zhang and Zuo, 2025; Ren, 2025; Lin et al., 2025; Luo et al., 2025a), while oth-

[1]While several reasoning methods integrate multiple strategies, each work is categorized based on its principal contribution for clarity.

ers construct high-quality training corpora (Liu et al., 2025c; Pei et al., 2025; Yu et al., 2025a).

(5) External Tool Integration augments LLM-based reasoning by incorporating external systems, such as logical solvers (Liu et al., 2025a; Wang et al., 2025; Yu et al., 2025b), program executors (Das et al., 2024), and knowledge retrievers (J. Liu et al., 2025; Yao and Yadav, 2025), to improve reasoning reliability.

(6) Verification methods incorporate post-generation validation to assess and refine reasoning outputs. Representative approaches evaluate answer consistency via masked prediction (Jiang et al., 2024a), regenerate and compare individual reasoning steps (Jiang et al., 2024b), or apply iterative step-level correction to improve reliability (Y. Li et al., 2025; Wu et al., 2025).

Despite strong empirical results, existing methods lack a unified and adaptive problem-solving framework. They mostly rely on one-size-fits-all, ad hoc strategies with fixed heuristics or pre-determined choices, failing to adapt reasoning operations to uncertain contexts or varying feedback. In response, this work proposes a fuzzy-guided reasoning framework that adaptively adopts actions based on intermediate LLM feedback.

Fuzzy Theory. Originating from (Zadeh, 1965), fuzzy theory models uncertainty through **fuzzy membership**. Formally, given a universe of discourse $\mathcal{X}$, a fuzzy set $\mathcal{F}$ is defined by a membership function $\mu_{\mathcal{F}} : \mathcal{X} \to [0, 1]$, which assigns each element a graded (membership) degree of belonging. Building on this representation, **fuzzy rules** translate membership values into interpretable and actionable inferences. A fuzzy rule typically takes the form:

$$\textbf{IF } \underbrace{x \text{ is } \mathcal{F}_x \textbf{ and } y \text{ is } \mathcal{F}_y}_{preconditions} \underbrace{\textbf{THEN } z \text{ is } \mathcal{F}_z}_{conclusion}, \quad (1)$$

where x, y, and z are (linguistic) variables, and $\mathcal{F}_{x/y/z}$ are associated fuzzy sets.

Fuzzy-based methods have been proven to be universal approximators of nonlinear mappings (Wang, 1992). They have since been adopted in a wide range of applications, including user profiling (Yang et al., 2022), translation (Hoang et al., 2023), information extraction and representation learning (Peng et al., 2023; Li et al., 2024; Xu et al., 2024a), and sentiment-related analysis (Li and Xiong, 2022; Fang et al., 2024; Jia et al., 2025; Chen et al., 2025). *Yet, their potential in mathematical reasoning remains largely unexplored.*

3 Methodology

This section presents the proposed **F**uzzy-**L**ogic-**A**ss**I**sted **R**easoner (FLAIR) for LLM mathematical reasoning, as illustrated in Fig. 3. We first introduce Fuzzy Reasoning Rules, followed by two key components of FLAIR: membership-based fuzzy inference and Reinforcement Learning (RL)–based rule weight updating.

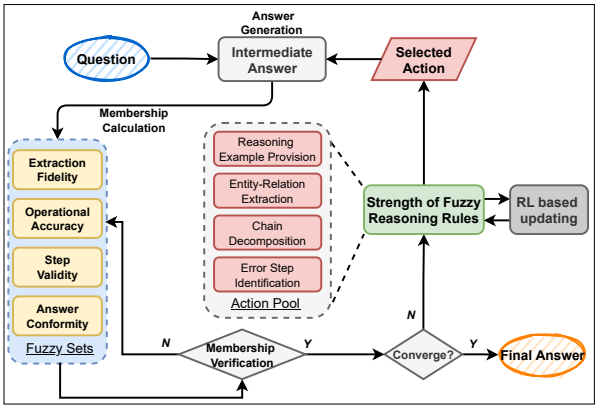


Figure 3: An overview of the proposed FLAIR for LLM mathematical reasoning, which is characterized by membership-driven fuzzy inference with Reinforcement Learning (RL)-based rule updating.

3.1 Fuzzy Reasoning Rules

FLAIR employs **F**uzzy **R**easoning **R**ules (FRRs) to determine appropriate actions. Following Eq. (1), each FRR consists of a set of *preconditions*, expressed as fuzzy sets that characterize the current problem-solving state, and an *action* as the rule conclusion. This formulation establishes a direct and interpretable mapping from fuzzy state assessments to actionable control decisions.

To begin with, the fuzzy sets used as preconditions are deliberately designed around potential reasoning failures. Their purpose is to provide a structured and uncertainty-aware summary of reasoning states, revealing the types and severities of deficiencies (if any) present at current stage, and further to enable the selective triggering of subsequent actions that directly address these potential failures. A systematic evaluation by Yin et al. (2025) examines the outputs of 15 LLMs across different scales and identifies 39 fine-grained errors. Building on this taxonomy, we further consolidate these errors into four categories, *i.e.*, *requirements*,

calculation, *reasoning*, and *answer*, which together account for 92.3% of total failures[2]. Accordingly, we define one fuzzy set for each of these categories as follows:

(a) **Extraction Fidelity set**($\mathcal{F}_{EF}$): indicates that the model accurately extracts all necessary information from the problem statement, corresponding to the *requirements* category.

(b) **Operational Accuracy set** ($\mathcal{F}_{OA}$): indicates that all arithmetic calculations in the reasoning chain are performed correctly, corresponding to the *calculation* category.

(c) **Step Validity set** ($\mathcal{F}_{SV}$): indicates that all intermediate reasoning steps are logically and methodologically valid, corresponding to the *reasoning* category.

(d) **Answer Conformity set** ($\mathcal{F}_{AC}$): indicates that the final answer is correct in terms of unit and format, corresponding to the *answer* category.

Together, these fuzzy sets form a unified state representation that captures reasoning deficiencies from complementary aspects. When used as preconditions in FRRs, they enable conditional activation of corrective actions tailored to diagnosed states.

Moreover, a set of corrective actions is introduced as rule conclusions, drawing from established reasoning strategies in prior work to address the reasoning deficiencies. Specifically, four actions are considered, including ($\mathcal{A}_1$) **Reasoning Example Provision**, ($\mathcal{A}_2$) **Entity-Relation Extraction**, ($\mathcal{A}_3$) **Chain Decomposition**, and ($\mathcal{A}_4$) **Error Step Identification**, each reflecting representative techniques summarized in Section 2. Notably, these actions are not introduced as novel techniques, but as modular components that can be conditionally activated within the proposed fuzzy framework. This design unifies diverse existing reasoning strategies under a shared fuzzy control mechanism. More details about implemented actions are provided in Appendix A.2.

Finally, combining fuzzy sets as preconditions with actions as rule conclusions yields the following FRR for LLM reasoning:

$$\begin{aligned} &\textbf{IF}\ \ x \text{ is } \mathcal{F}_{EF} \textbf{ and } x \text{ is } \mathcal{F}_{OA} \\ &\quad\ \textbf{and } x \text{ is } \mathcal{F}_{SV} \textbf{ and } x \text{ is } \mathcal{F}_{AC} \\ &\textbf{THEN}\ \ \text{Action is } \mathcal{A}_i, \end{aligned} \tag{2}$$

where x denotes the current output from LLM. The proposed FRR jointly considers all fuzzy sets as preconditions and determines rule activation based on membership degrees (*i.e.*, the set belongings). This design avoids rigid error–action coupling and enables flexible fuzzy control under overlapping or mixed reasoning deficiencies. As a result, actions are conditionally triggered by fuzzy membership degrees, *without assuming predefined correspondences between errors and corrective actions*.

Importantly, the proposed FLAIR is inherently extensible: additional fuzzy sets and actions can be incorporated by expanding the rule pool, which is left for future exploration.

3.2 Fuzzy inference

The aforementioned FRRs are leveraged to steer the LLM reasoning via a three-stage workflow: step-wise generation, membership calculation and verification, and solution revision.

(1) Step-wise Generation. The reasoning process is executed in multiple steps. At each step, an *answer-generation LLM* is prompted to produce a structured intermediate solution, which consists of extracted conditions, intermediate calculations, and a provisional answer. See Appendix B.1 for prompt details.

(2) Membership Calculation and Verification. The intermediate (step-wise) answer is then subjected to fuzzy verification using predefined fuzzy sets and their memberships (μ) to diagnose current states. *A higher membership μ indicates a lower risk of the corresponding error type, and vice versa.* For example, $\mu_{EF} = 0.9$ suggests a low likelihood of a requirement error, whereas $\mu_{OA} = 0.2$ indicates a high probability of a calculation error. In FLAIR, memberships are computed by prompting a *membership-calculation LLM* with explicit instructions, as detailed in Appendix B.2. Alternative membership computation schemes are further examined in the ablation study.

To validate the estimated memberships, a *verifier LLM* is further prompted using a small set of reference samples (*i.e.*, in a few-shot setting) to re-compute the membership degrees and provide rationales[3]. If re-evaluated memberships deviate from original estimates beyond a threshold (λ_{ver}), the memberships are considered unreliable and re-estimated by the membership calculator (with the verifier's rationale). Alternative verification strategies are examined in the ablation study.

[2]The detailed mapping from the error types in (Yin et al., 2025) to our four categories is provided in Appendix A.1.

[3]Details of the reference samples and membership verification prompts are provided in Appendix C and B.3, respectively.

(3) Solution Revision. If all estimated membership degrees exceed a predefined threshold (λ_{rvs}), the current solution is accepted. Otherwise, the firing strengths of FRRs are evaluated. Let N_a denote the number of actions and N_{FS} the number of fuzzy sets. Each rule is parameterized by learnable weights β_{ij}, indicating the contribution of the j-th fuzzy set to the i-th rule ($i \in [1, N_a]$; $j \in [1, N_{FS}]$). Given the membership μ_j of the j-th fuzzy set, the firing strength of the i-th rule can be computed as follows:

$$\sigma_i = \sum_{j=1}^{N_{FS}} \beta_{ij}\, \mu_j \quad (\forall i \in [1, N_a]). \tag{3}$$

The rule with maximal σ_i is selected, and its associated action is applied to refine current solution. Detailed prompts for employed actions are provided in Appendix B.4.

3.3 RL-based updating

To improve reasoning effectiveness, an RL-based method is further introduced to optimize rule weights. Let $\mathbf{B} = [\beta_{ij}] \in \mathbb{R}^{N_a \times N_{FS}}$ denote the rule-weight matrix. Given $\{\mu_j(t)\}_{j=1}^{N_{FS}}$ at step t, the following reward function is adopted:

$$R(t) = \alpha \left(\min_{1 \le j \le N_{FS}} \mu_j(t) \right)^2 + (1-\alpha)\, \log\det\Big(\mathbf{B}(t)^\top \mathbf{B}(t) + \epsilon \mathbf{I} \Big), \tag{4}$$

where $\det(\cdot)$ denotes the matrix determinant, $\epsilon = 10^{-4}$, and α is the hyperparameter. The first term encourages high-confidence memberships across fuzzy sets, while the second term promotes diversity among the rule weights (Kulesza and Taskar, 2012). $R(t)$ is then used to update $\mathbf{B}(t)$ via the Proximal Policy Optimization (PPO) algorithm.

Remark 1. *In* FLAIR*, each iteration follows a generation–verification–revision cycle, while generated fuzzy memberships are used for action activation and RL-based reward computation. The process terminates either the predefined membership threshold (λ_{rvs}) is satisfied or the maximum of reasoning steps (t_{max}) is reached.*

Remark 2. *The proposed reasoning process can operate directly on test instances, as rule weights are optimized using feedback derived solely from the model outcome, without external supervision (such as ground-truth answers). This setting is conceptually related to* ***test-time training****, in that adaptation is guided by outcome-based feedback during inference. Interested readers can explore supervised learning for fuzzy reasoning rules.*

4 Experiments

4.1 Experimental Setup

Datasets and Base LLMs. Evaluation is conducted on five widely-used mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), SVAMP (Patel et al., 2021), AMC (AI-MO Team, 2024b), and AIME (AI-MO Team, 2024a). These datasets cover diverse reasoning skills and problem difficulties, including arithmetic computation, logical reasoning, and multi-step problem solving.

Baselines. Performance is evaluated in comparison to 17 state-of-the-art reasoning methods, including LBS3 (Luo et al., 2025b), StepCo (Wu et al., 2025), and SAFE (Liu et al., 2025a), among others. Detailed descriptions of all baselines are provided in Appendix G. In addition, the few-shot performance of vanilla LLMs is also reported as a reference baseline. All baseline results are directly sourced from the original papers.

Implementation. For FLAIR, six LLMs are adopted as base models, including both closed-source models (GPT-4 and GPT-4o) and open-source models (Llama2/3(-70B), DeepSeek(Math-7B), and Qwen2(-72B). The same base LLM is used for answer generation, membership calculation and verification, while an ablation study examining alternative LLM for these components is provided in Appendix D. In addition, adopted hyperparameters are set to $t_{\max} = 5$, $\alpha = 0.8$, $\lambda_{\text{ver}} = 0.3$ and $\lambda_{\text{rvs}} = 0.9$. Final answer accuracy is adopted as the evaluation metric.

4.2 Main Results

Table 1 and 2 present the detailed performance comparison across five benchmarks and six base LLMs on five runs. The experimental results demonstrate that FLAIR consistently achieves superior or competitive performance across all five datasets. For instance, FLAIR obtains the best performance in 13 out of 22 evaluation settings, while achieving 8 second-best results in the remaining cases.

Notably, as an unsupervised method, FLAIR delivers performance that is on par with, and in several cases exceeds, fully supervised reasoning methods (*e.g.*, on GSM8K with LLaMA3 and on

Table 1: Performance comparison of various reasoning methods across GSM8K, MATH, and SVAMP datasets. * denotes unsupervised methods, while † denotes supervised methods. FS-CoT denotes few-shot chain-of-thought prompting using a vanilla LLM. Results marked with (a) are evaluated on MATH500 (a subset of MATH), and results marked with (b) are evaluated on Qwen1.5-72B. The **best** and second-best results are highlighted.

Methods	GSM8K	MATH	SVAMP
GPT-4			
FS-CoT*	90.3	48.9	92.9
FOBAR*	96.4 ↑ 6.1	-	94.4 ↑ 1.5
LBS3*	94.9 ↑ 4.6	64.2 ↑ 15.3	93.5 ↑ 0.6
Self-Contrast*	95.4 ↑ 5.1	-	94.0 ↑ 1.1
FLAIR *	**98.4** ↑ 8.1	**82.2** ↑ 33.3	**97.0** ↑ 4.1
±Std.	0.1	0.4	0.2
Llama2			
FS-CoT*	70.4	20.2	60.1
Self-Contrast*	64.2 ↓ 6.2	-	75.3 ↑ 15.2
WizardMath†	**92.8** ↑ 22.4	**58.6** ↑ 38.4	-
MetaMath†	82.3 ↑ 11.9	26.6 ↑ 6.4	-
FLAIR *	88.1 ↑ 17.7	32.4 ↑ 12.2	**86.3** ↑ 26.2
±Std.	1.2	2.5	0.8
DeepSeek			
FS-CoT*	85.3	40.8	70.8
SAFE*	87.6 ↑ 2.3	52.4 ↑ 11.6	-
MathFusion†	77.9 ↓ 7.4	53.4 ↑ 12.6	-
WizardMath†	**91.6** ↑ 6.3	**64.6** ↑ 23.8	-
DART†	88.2 ↑ 2.9	53.6 ↑ 12.8	-
MMIQC†	79.0 ↓ 6.3	45.3 ↑ 4.5	-
KP†	83.9 ↓ 1.4	48.8 ↑ 8.0	-
FLAIR *	90.3 ↑ 5.0	56.4 ↑ 15.6	**77.3** ↑ 6.5
±Std.	0.5	0.9	1.3

Methods	GSM8K	MATH	SVAMP
GPT-4o			
FS-CoT*	95.0	76.9	93.3
SAFE*	96.0 ↑ 1.0	80.4 ↑ 3.5	-
StepCo†	**96.4** ↑ 1.4	80.4$^{(a)}$ ↑ 3.5	96.0 ↑ 2.7
EAP†	-	79.2 ↑ 2.3	-
FLAIR *	96.1 ↑ 1.1	**81.2** ↑ 4.3	**96.9** ↑ 3.6
±Std.	0.2	0.6	0.4
Llama3			
FS-CoT*	94.2	26.1	76.3
MultiTAG*	-	**84.2**$^{(a)}$ ↑ 58.1	-
LBS3*	94.6 ↑ 0.4	59.6 ↑ 33.5	93.6 ↑ 17.3
SKTO†	-	79.6$^{(a)}$ ↑ 53.5	-
MCoT†	83.1 ↓ 11.1	54.7 ↑ 28.6	-
DART†	90.4 ↓ 3.8	56.1 ↑ 30.0	-
MMIQC†	89.3 ↓ 4.9	49.4 ↑ 23.3	-
FLAIR *	**96.2** ↑ 2.0	64.6 ↑ 38.5	**96.5** ↑ 20.2
±Std.	0.2	1.4	0.3
Qwen2			
FS-CoT*	93.9	72.2	90.5
SGR*	-	79.2 ↑ 7.0	-
LBS3*	88.8$^{(b)}$ ↓ 5.1	53.1$^{(b)}$ ↓ 19.1	91.0$^{(b)}$ ↑ 0.5
AceMath†	**96.4** ↑ 2.5	**84.5** ↑ 12.3	-
FLAIR *	**96.4** ↑ 2.5	81.2 ↑ 9.0	**95.1** ↑ 4.6
±Std.	0.4	0.7	0.5

Table 2: Performance comparison of various reasoning methods on AMC and AIME datasets. * indicates unsupervised methods, while † indicates supervised methods. The **best** and second-best results are highlighted.

Methods	AMC	AIME
LLaMA3		
FS-CoT*	40.0	13.3
MultiTAG*	67.5 ↑ 27.5	38.9 ↑ 25.6
SKTO†	**70.0** ↑ 30.0	30.0 ↑ 16.7
FLAIR *	68.8 ↑ 28.8	**46.7** ↑ 33.4
±Std.	1.0	3.3
Qwen2		
FS-CoT*	35.0	6.7
SGR*	**61.3** ↑ 26.3	8.0 ↑ 1.3
FLAIR *	55.0 ↑ 20.0	**16.7** ↑ 10.0
±Std.	2.5	1.7

MATH with GPT-4o). This observation indicates that effective reasoning control can be achieved through adaptive and uncertainty-aware fuzzy regulation. Compared with existing six unsupervised approaches, FLAIR achieves the best performance on average. Although its performance is lower in specific case (*e.g.*, on MATH with Llama3), this reason is largely due to the use of external tools from its peer (*i.e.*, MultiTAG), whereas FLAIR operates solely through LLM prompting.

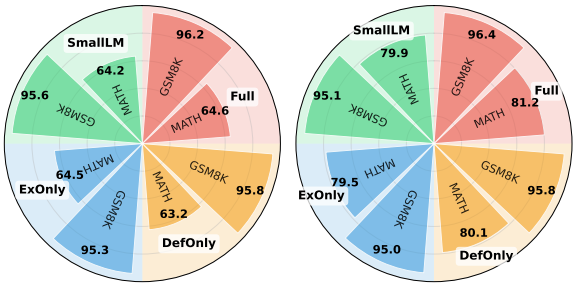
Overall, the proposed FLAIR consistently delivers competitive and often superior performance across diverse benchmarks. Its effectiveness is primarily attributed to the ability to characterize reasoning uncertainty through proposed fuzzy sets and associated membership, which in turn enables more adaptive fuzzy reasoning rules and leads to more accurate reasoning behavior.

5 Ablation Study

To analyze the effectiveness of FLAIR, a series of ablation studies is conducted on GSM8K and MATH, which represent varying levels of problem complexity (Yu et al., 2025a). The LLaMA3(-70B) and Qwen2(-72B) are selected as the base LLMs.

Membership calculation. This subsection examines the impact of different membership-calculation strategies on accuracy. In main experiments, memberships are estimated using prompts with both fuzzy-set definitions and illustrative examples (labeled as Full), while details about these examples can be found in Appendix C. To compare, several variants are evaluated: (i) DefOnly uses prompts containing only fuzzy-set definitions; (ii) ExOnly relies solely on illustrative examples; and (iii) SmallLM employs a separately trained

lightweight model to estimate memberships[4].



(a) LLaMA3-base (b) Qwen2-base

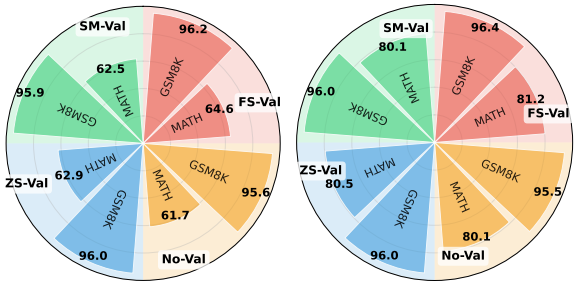
Figure 4: Impact from three alternative membership-calculation strategies.

The results in Fig. 4 show that removing either component from the calculation prompt consistently degrades performance, indicating that conceptual definitions and illustrative examples provide complementary information. For example, compared to `Full`, the `DefOnly` and `ExOnly` variants reduce accuracy on `GSM8K` by 0.4% and 0.9% for LLaMA3, and by 0.6% and 1.4% for Qwen2, respectively. We also notice that, the `ExOnly` variant exhibits a larger performance drop (compared to `DefOnly`), which is likely due to limited alignment between provided examples and the target problem, as examples alone cannot comprehensively convey the underlying conceptual principles. In addition, the `SmallLM` variant achieves slightly better performance comparable to `ExOnly`. This observation, again, is potentially due to its training samples, which provides partial but insufficient support for generalizing across diverse problems.

Membership validation. This subsection analyzes the effect of the membership-validation step to overall performance. In the main setting, membership values are validated through consistency checking against annotated reference samples, referred to as the `FS-Val` method. To assess its effect, three additional variants are considered: (i) `No-Val` removes validation entirely, (ii) `ZS-Val` performs validation via a zero-shot manner, and (iii) `SM-Val` employs a smaller model (LLaMA-7B) as the verifier in place of the base LLM.

As shown in Fig. 5, removing validation (`No-Val`) consistently degrades performance, yielding drops of 0.6% and 0.9% on `GSM8K` for LLaMA3 and Qwen2, respectively, and larger declines on `MATH` (2.9% for LLaMA3 and 1.1% for Qwen2).

[4]Details of the lightweight model and its training procedure are provided in Appendix E.



(a) LLaMA3-base (b) Qwen2-base

Figure 5: Impact from three alternative membership-validation strategies.

Meanwhile, `SM-Val` remains competitive compared with `No-Val`. This indicates that example-guided validation supports more accurate inference, and the fuzzy control mechanism is robust to verifier capacity.

Effect of actions. This subsection studies the contribution of individual actions (or rules) by removing one at a time while keeping others unchanged. As shown in Fig. 6, all actions contribute positively to the overall performance, as removing any individual action consistently leads to performance degradation. However, the relative importance of different actions varies across datasets with different reasoning difficulties.

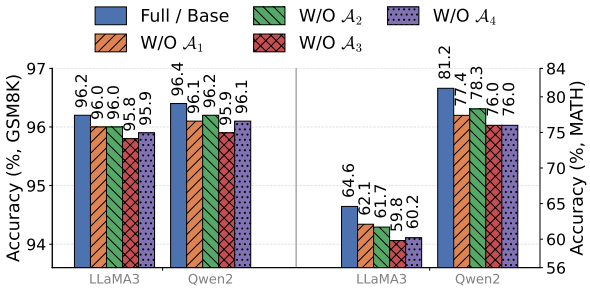


Figure 6: Effect of adopted actions on answering accuracy with GSM8K and MATH under two LLMs.

On average, excluding $\mathcal{A}_3$ and $\mathcal{A}_4$ results in a larger accuracy drop than excluding $\mathcal{A}_1$ and $\mathcal{A}_2$. This effect is particularly observed on the `MATH` dataset, where removing either $\mathcal{A}_3$ or $\mathcal{A}_4$ causes larger performance degradation, reflecting the increased complexity and longer reasoning chains required by this benchmark. These observations highlight the critical role of explicit chain decomposition and error step identification in complex mathematical reasoning. Such findings are consistent with prior work (Leang et al., 2025; Xu et al., 2025) for solving challenging mathematical problems. In contrast, removing $\mathcal{A}_1/\mathcal{A}_2$ leads to a moderate yet

consistent performance decline across datasets, in particular with the MATH dataset. Overall, these results confirm that, within the proposed FLAIR framework, different actions play complementary roles depending on task difficulty, demonstrating that fuzzy control enables adaptive action selection and robust reasoning capability.

Alternative fuzzy set design. In the main experiments, fuzzy sets are manually designed based on established reasoning failure categories. To examine whether FLAIR generalizes beyond this specific design choice, we implement a data-driven alternative based on skill-aware clustering (Didolkar et al., 2024). Specifically, we prompt the LLM to annotate each training instance with its problem type and required reasoning skills, then apply k-means clustering to group instances accordingly, where each cluster serves as a candidate fuzzy set. This modification only affects the definition of fuzzy sets, while the overall FLAIR framework remains unchanged. We evaluate this variant under different numbers of clusters K on GSM8K and MATH using GPT-4 as the backbone, and report results alongside the manually designed configuration in Table 3.

As observed, automatically discovered fuzzy sets are effective and stable. Performance improves as K increases, reaching its peak at K=9 on GSM8K (98.7%) and at K=13 on MATH (82.9%), while excessively large K leads to performance degradation. Importantly, the comparable performance across both configurations confirms that the effectiveness of FLAIR does not hinge on handcrafted fuzzy sets. Instead, FLAIR can adapt to other reasoning domains through data-driven fuzzy partitioning.

Table 3: Performance comparison between automatically discovered (k-means) and manually designed fuzzy sets on GSM8K and MATH using GPT-4.

Setting	GSM8K	MATH
$K = 3$	85.1	45.7
$K = 5$	91.5	64.2
$K = 7$	94.9	77.6
$K = 9$	**98.7**	81.8
$K = 11$	98.4	82.4
$K = 13$	97.9	**82.9**
$K = 15$	96.8	82.5
Manual 4-set (FLAIR)	98.4	82.2

Reward analysis. This subsection analyzes the hyperparameter α from Eq. (4). Specifically, α is varied over $\{1, 0.8, 0.6, 0.4, 0.2, 0\}$ to examine its influence, while others are kept unchanged. As observed from Fig. 7, the model accuracy initially improves as α decreases, reaching its peak at intermediate values (*e.g.*, $\alpha = 0.6$ or 0.8). However, as α further decreases, the accuracy gradually degrades, reaching its minimum when $\alpha = 0$. This degradation could be attributed to the insufficient emphasis on fuzzy membership, as more weight shifted toward rule diversity (with smaller α), which in turn deteriorates the quality of fuzzy inference and weakens the effectiveness of reasoning regulation.

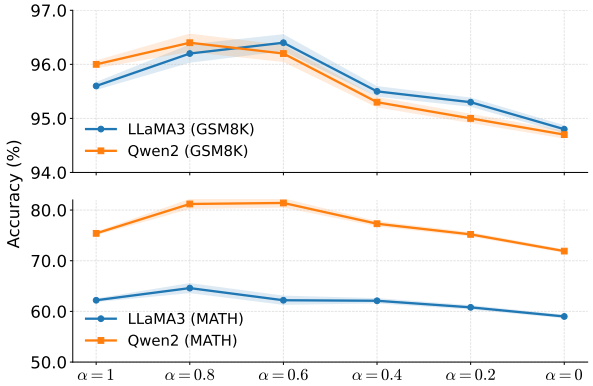


Figure 7: Hyperparameter α sensitivity analysis.

Rule weight updating strategy. To explicitly examine the contribution of RL-based rule weight updating, we compare FLAIR against two alternative weighting strategies: (i) **Uniform**, where all rule weights are set to a constant value (in our case, $w = 0.25$); and (ii) **Softmax**, where rule weights are parameterized and normalized via a softmax function over learnable logits, but without reward-driven optimization.

As shown in Table 4, RL-based updating consistently outperforms both alternatives, with more performance improvements observed on the complex MATH benchmark. Notably, simple parameterization (**Softmax**) already introduces learnable flexibility, yet still falls short of reward-driven optimization. This indicates that the performance gain is attributed to adaptive, reward-guided policy learning, rather than from arbitrary weighting or increased parameterization.

Table 4: Impact analysis on RL-based rule weight updating strategies using GPT-4.

Setting	GSM8K	MATH
Uniform	97.3	79.6
Softmax	97.8	80.8
Ours	**98.4**	**82.2**

Computational efficiency. Since FLAIR relies on the LLM prompting, we further evaluate its com-

putational efficiency by comparing the number of API calls with existing methods.

For instance, with **SGR** (Cao et al., 2025), in the first iteration, three API calls are required to generate (1) the step-guidance question, (2) the guidance itself, and (3) the initial step answer. For each subsequent iteration, two additional calls are made to produce the next guidance cue and its refined response. Overall, the total number of API calls is $3 + 2(T - 1)$ for T iterations. For **Self-Refine** (Madaan et al., 2023), in the first iteration, one API call is required to generate the initial solution. For each subsequent iteration, two additional calls are made to produce (1) the self-feedback critique and (2) the refined response. Overall, the total number of API calls is $1 + 2(T - 1) = 2T - 1$ for T iterations. For **Self-Contrast** (Zhang et al., 2024b), in the first stage, $n + 1$ API calls are required to generate (1) n candidate answer perspectives and (2) the corresponding initial solutions. In the subsequent stage, two additional calls are made to produce (1) a structural checklist for those solutions and (2) the final refined response. Overall, the total number of API calls is $n + 3$ where n typically ranges from 2 to 5. For **LBS3** (Luo et al., 2025b), in the initial phase, one API call is required to generate the so-called n_e easy and n_h hard proxy queries. In the subsequent phase, n_e additional calls are made to solve each easy proxy query using zero-shot CoT, followed by n_h calls to solve each hard proxy query using few-shot CoT conditioned on the previously solved easy examples. Finally, one additional call is used to produce the final solution. Overall, the total number of API calls is $1+n_e+n_h+1$, where for GSM8K, n_e+n_h is fixed at 3, and for MATH, $n_e + n_h$ is fixed at 4 in their paper. For **SAFE** (Liu et al., 2025a), in the initial generation phase, n API calls are required to sample n candidate reasoning trajectories using zero-shot CoT from the LLM. Each trajectory is then decomposed into m reasoning steps. For every step, an average of a additional API calls are made to formalize the step into a Lean statement. Overall, the total number of API calls is $n \times (1 + m \times a)$.

Overall, FLAIR requires 3 API calls per iteration, yet achieves comparable or even superior performance compared to existing approaches. In addition, empirical comparisons, including total token usage and tokens per API call, are also provided; further details can be found in Appendix F. The experimental results are consistent with the aforementioned analysis, demonstrating its superiority.

6 Case study

A case study (from GSM8K, Test ID: 855) is provided in Fig. 8, which illustrates how FLAIR incrementally guides the model toward a correct solution through adaptive diagnosis and correction.

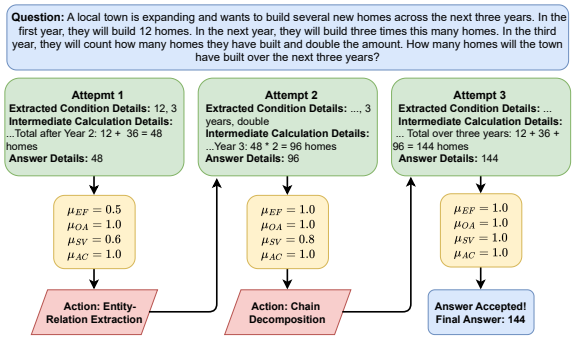


Figure 8: An illustrating example about the process of error detection and answer refinement in FLAIR.

At the initial attempt, the LLM produces an incorrect answer due to incomplete conditions, leading to low membership scores in extraction fidelity and step validity. To address, two actions are sequentially activated, *i.e.*, the *Entity-Relation Extraction* action in Attempt 1 and the *Chain Decomposition* action in Attempt 2. Incorporating these actions, the LLM generates a revised solution that attains improved membership scores. Consequently, the resulting answer satisfies the acceptance criteria and is returned as the final output.

7 Conclusion

This paper introduces FLAIR, a fuzzy-guided framework for adaptive LLM-based mathematical reasoning. To the best of our knowledge, this work represents the first systematic integration of fuzzy theory into LLM mathematical reasoning. By modeling intermediate problem-solving states through fuzzy memberships and regulating corrective actions via fuzzy reasoning rules, the proposed approach provides a fuzzy mechanism for handling uncertainty and overlapping reasoning deficiencies.

Extensive experiments across multiple LLMs and benchmarks demonstrate that FLAIR consistently improves reasoning accuracy and reliability. Beyond mathematical problem solving, the proposed framework is task-agnostic and can be readily extended to other domains, such as agent control, knowledge distillation, and tool-augmented task planning, where uncertainty-aware and adaptive control is crucial.

Limitations

This work is the first attempt to use fuzzy theory to LLM reasoning. Despite its effectiveness, the proposed framework has several limitations. First, the design of fuzzy sets, membership calculation and verification strategies, and fuzzy reasoning rules currently relies on a predefined configuration tailored to common reasoning errors. While this design is flexible and extensible, incorporating additional fuzzy sets or alternative rule structures will further improve coverage for more complex or domain-specific reasoning cases.

Second, the current implementation adopts relatively simple fuzzy inference and aggregation mechanisms. More advanced fuzzy theories, such as type-2 fuzzy sets or hierarchical fuzzy systems, may offer richer uncertainty modeling and finer-grained reasoning control. Exploring these extensions, as well as automated discovery of fuzzy sets and rules, constitutes an important direction for future work.

Finally, although the proposed framework operates in a test-time adaptive manner, its computational overhead increases with iterative refinement. Balancing reasoning performance and efficiency remains an open challenge, motivating the development of more effective and efficient Reinforcement Learning strategies for adaptive reasoning control.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable suggestions to improve the quality of the article.

The work of Jie Yang is supported by Australian Research Council Discovery Project (DP210101426). The work of Xinguo Yu and Hao Wu are supported by General Project of the National Natural Science Foundation of China (Grant No: 62277022). The work of Hao Wu is also supported by scholarship from the China Scholarship Council.

References

AI-MO Team. 2024a. AIMO Validation AIME dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AI-MO Team. 2024b. AIMO Validation AMC dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Lang Cao, Yingtian Zou, Chao Peng, Renhong Chen, Wu Ning, and Yitong Li. 2025. Step Guided Reasoning: Improving Mathematical Reasoning using Guidance Generation and Step Reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21112–21129, Suzhou, China. Association for Computational Linguistics.

Ping Chen, Xiang Liu, Zhaoxiang Liu, Zezhou Chen, Xingpeng Zhang, Huan Hu, Zipeng Wang, Kai Wang, Shuming Shi, and Shiguo Lian. 2025. Fuzzy reasoning chain (FRC): An innovative reasoning framework from fuzziness to clarity. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10230–10240, Suzhou, China. Association for Computational Linguistics.

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

Debrup Das, Debopriyo Banerjee, Somak Aditya, and Ashish Kulkarni. 2024. MATHSENSEI: A tool-augmented large language model for mathematical reasoning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 942–966, Mexico City, Mexico. Association for Computational Linguistics.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1330–1350, Bangkok, Thailand. Association for Computational Linguistics.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. 2024. Metacognitive Capabilities of LLMs: An Exploration in Mathematical Problem Solving. In *Advances in Neural Information Processing Systems*, volume 37, pages 19783–19812. Curran Associates, Inc.

Yang Fang, Cheng Xu, Shuhao Guan, Nan Yan, and Yuke Mei. 2024. Advancing Arabic sentiment analysis: ArSen benchmark and the improved fuzzy deep hybrid network. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 507–516, Miami, FL, USA. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Cuong Hoang, Devendra Sachan, Prashant Mathur, Brian Thompson, and Marcello Federico. 2023. Improving retrieval augmented neural machine translation by controlling source and fuzzy-match interactions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 289–295, Dubrovnik, Croatia. Association for Computational Linguistics.

Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, Fan Yang, and Mao Yang. 2024. Fewer is more: Boosting math reasoning with reinforced context pruning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13674–13695, Miami, Florida, USA. Association for Computational Linguistics.

Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. 2025. Key-point-driven data synthesis with its enhancement on mathematical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24176–24184.

J. Liu, Z. Huang, Q. Liu, Z. Ma, C. Zhai, and E. Chen. 2025. Knowledge-Centered Dual-Process Reasoning for Math Word Problems With Large Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 37(6):3457–3471.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. 2025. The task shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29680–29697, Vienna, Austria. Association for Computational Linguistics.

Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James Kwok. 2024a. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6647–6661, Bangkok, Thailand. Association for Computational Linguistics.

Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li, and Dongsheng Li. 2024b. Llms can find mathematical reasoning mistakes by pedagogical chain-of-thought. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3439–3447. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286.

Joshua Ong Jun Leang, Aryo Pradipta Gema, and Shay B Cohen. 2025. CoMAT: Chain of mathematically annotated thought improves mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20256–20285, Suzhou, China. Association for Computational Linguistics.

Junzhuo Li and Deyi Xiong. 2022. KaFSP: Knowledge-aware fuzzy semantic parsing for conversational question answering over a large-scale knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 461–473, Dublin, Ireland. Association for Computational Linguistics.

Yiyuan Li, Shichao Sun, and Pengfei Liu. 2024. FRoG: Evaluating fuzzy reasoning of generalized quantifiers in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7239–7256, Miami, Florida, USA. Association for Computational Linguistics.

Zhenwen Liang, Jipeng Zhang, and Xiangliang Zhang. 2023. Don't be blind to questions: Question-oriented math word problem solving. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15–25, Nusa Dua, Bali. Association for Computational Linguistics.

Yen-Ting Lin, Di Jin, Tengyu Xu, Tianhao Wu, Sainbayar Sukhbaatar, Chen Zhu, Yun He, Yun-Nung Chen, Jason E Weston, Yuandong Tian, Arash Rahnama, Sinong Wang, Hao Ma, and Han Fang. 2025. Step-KTO: Optimizing Mathematical Reasoning through Stepwise Binary Feedback. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 15–33, Suzhou, China. Association for Computational Linguistics.

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. 2025a. Safe: Enhancing Mathematical Reasoning in Large Language Models via Retrospective Step-aware Formal Verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12171–12186, Vienna, Austria. Association for Computational Linguistics.

Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew C Yao. 2025b. Augmenting math word problems via iterative question composing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24605–24613.

Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025c. AceMath: Advancing Frontier Math Reasoning with Post-Training and Reward Modeling. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3993–4015, Vienna, Austria. Association for Computational Linguistics.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. 2025a. WizardMath: Empowering Mathematical Reasoning for Large Language Models via Reinforced Evol-Instruct. In *The Thirteenth International Conference on Learning Representations*.

Kangyang Luo, Zichen Ding, Zhenmin Weng, Lingfeng Qiao, Meng Zhao, Xiang Li, Di Yin, and Jinlong Shu. 2025b. Let's be self-generated via step by step: A curriculum learning approach to automated reasoning with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15343–15420, Vienna, Austria. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Qizhi Pei, Lijun Wu, Zhuoshi Pan, Yu Li, Honglin Lin, Chenlin Ming, Xin Gao, Conghui He, and Rui Yan. 2025. MathFusion: Enhancing Mathematical Problem-solving of LLM through Instruction Fusion. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7400–7420, Vienna, Austria. Association for Computational Linguistics.

Tianshuo Peng, Zuchao Li, Lefei Zhang, Bo Du, and Hai Zhao. 2023. FSUIE: A novel fuzzy span mechanism for universal information extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16318–16333, Toronto, Canada. Association for Computational Linguistics.

Syed Rifat Raiyan, Md Nafis Faiyaz, Shah Md. Jawad Kabir, Mohsinul Kabir, Hasan Mahmud, and Md Kamrul Hasan. 2023. Math word problem solving by generating linguistic variants of problem statements. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 362–378, Toronto, Canada. Association for Computational Linguistics.

Hangliang Ren. 2025. LSRL: Process-Supervised GRPO on Latent Recurrent States Improves Mathematical Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12534–12545, Suzhou, China. Association for Computational Linguistics.

Evangelia Spiliopoulou, Riccardo Fogliato, Hanna Burnsky, Tamer Soliman, Jie Ma, Graham Horwood, and Miguel Ballesteros. 2025. Play favorites: A statistical method to measure self-bias in llm-as-a-judge. *Preprint*, arXiv:2508.06709.

Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846.

L. X. Wang. 1992. Fuzzy systems are universal approximators. *International Journal of General Systems*, 17(2-3):59–75.

Ruida Wang, Yuxin Li, Yi R. Fung, and Tong Zhang. 2025. Let's Reason Formally: Natural-Formal Hybrid Reasoning Enhances LLM's Math Capability. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16794–16820, Suzhou, China. Association for Computational Linguistics.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2025. Enhancing Mathematical Reasoning in LLMs by Stepwise Correction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21602–21623, Vienna, Austria. Association for Computational Linguistics.

Fred Xu, Song Jiang, Zijie Huang, Xiao Luo, Shichang Zhang, Yuanzhou Chen, and Yizhou Sun. 2024a. FUSE: Measure-theoretic compact fuzzy set representation for taxonomy expansion. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2707–2720, Bangkok, Thailand. Association for Computational Linguistics.

Jundong Xu, Hao Fei, Meng Luo, Qian Liu, Liangming Pan, William Yang Wang, Preslav Nakov, Mong-Li Lee, and Wynne Hsu. 2025. Aristotle: Mastering logical reasoning with a logic-complete decompose-search-resolve framework. In *Proceedings of the*

63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3052–3075, Vienna, Austria. Association for Computational Linguistics.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024b. Pride and prejudice: LLM amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15474–15492, Bangkok, Thailand. Association for Computational Linguistics.

Y. Li, D. M. Ubaidali, L. Wang, and W. Zhang. 2025. Step-by-Step Correction of LLM-based Math Word Problems Solutions. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. Journal Abbreviation: ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Yibo Yan, Jiamin Su, Jianxiang He, Fangteng Fu, Xu Zheng, Yuanhuiyi Lyu, Kun Wang, Shen Wang, Qingsong Wen, and Xuming Hu. 2025. A survey of mathematical reasoning in the era of multimodal large language model: Benchmark, method & challenges. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11798–11827, Vienna, Austria. Association for Computational Linguistics.

Jie Yang, Brian Yecies, Jun Ma, and Wanqing Li. 2022. Sparse fuzzy classification for profiling online users and relevant user-generated content. *Expert Systems with Applications*, 194:116497.

Wen Yang, Minpeng Liao, and Kai Fan. 2025. Markov Chain of Thought for Efficient Mathematical Reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7132–7157, Albuquerque, New Mexico. Association for Computational Linguistics.

Bohan Yao and Vikas Yadav. 2025. Diverse Multi-tool Aggregation with Large Language Models for Enhanced Math Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 25264–25282, Suzhou, China. Association for Computational Linguistics.

Zhangyue Yin, YuHong Sun, Xuanjing Huang, Xipeng Qiu, and Hui Zhao. 2025. Error Classification of Large Language Models on Math Word Problems: A Dynamically Adaptive Framework. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 338–365, Suzhou, China. Association for Computational Linguistics.

Erxin Yu, Jing Li, Ming Liao, Qi Zhu, Boyang Xue, Minghui Xu, Baojun Wang, Lanqing Hong, Fei Mi, and Lifeng Shang. 2025a. Self-Error-Instruct: Generalizing from Errors for LLMs Mathematical Reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8504–8519, Vienna, Austria. Association for Computational Linguistics.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Yiyao Yu, Yuxiang Zhang, Dongdong Zhang, Xiao Liang, Hengyuan Zhang, Xingxing Zhang, Mahmoud Khademi, Hany Hassan Awadalla, Junjie Wang, Yujiu Yang, and Furu Wei. 2025b. Chain-of-Reasoning: Towards Unified Mathematical Reasoning in Large Language Models via a Multi-Paradigm Perspective. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24914–24937, Vienna, Austria. Association for Computational Linguistics.

L. A. Zadeh. 1965. Fuzzy sets. *Information and Control*, 8(3):338–353.

Jixiao Zhang and Chunsheng Zuo. 2025. GRPO-LEAD: A Difficulty-Aware Reinforcement Learning Approach for Concise Mathematical Reasoning in Language Models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5642–5665, Suzhou, China. Association for Computational Linguistics.

Wenqi Zhang, Yongliang Shen, Guiyang Hou, Kuangyi Wang, and Weiming Lu. 2024a. Specialized mathematical solving by a step-by-step expression chain generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3128–3140.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024b. Self-contrast: Better reflection through inconsistent solving perspectives. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3602–3622, Bangkok, Thailand. Association for Computational Linguistics.

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 *The Twelfth International Conference on Learning Representations*.

Yue Zhou, Yada Zhu, Diego Antognini, Yoon Kim, and Yang Zhang. 2024. Paraphrase and solve: Exploring and exploiting the impact of surface form on mathematical reasoning in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2793–2804, Mexico City, Mexico. Association for Computational Linguistics.

A Implementation of Fuzzy Reasoning Rules

This section details the implementation of *Fuzzy Reasoning Rules* (FRRs), focusing on two core aspects: (1) the construction of fuzzy **preconditions** that diagnose the model's reasoning state (see Section A.1), and (2) the formulation of rule-triggered actions (*i.e.*, **conclusions**) that adaptively activated in the reasoning process (see Section A.2).

A.1 Error Taxonomy (Rule Precondition)

A systematic large-scale analysis by Yin et al. (2025) evaluates 15 LLMs of varying scales across four mathematical reasoning benchmarks and identifies 304,685 erroneous outputs, which are further classified into 39 fine-grained error types.

Building on this taxonomy, a *many-to-one* consolidation strategy is adopted to group these fine-grained errors into four higher-level categories: *requirements*, *calculation*, *reasoning*, and *answer*, as summarized in Table 5. These four categories cover 36 out of the original 39 error types, including all ten most frequent errors reported in Yin et al. (2025), and collectively account for 92.3% of the observed failures.

These higher-level categories are further employed as *rule preconditions* for reasoning-state diagnosis, serving as the semantic basis for constructing fuzzy sets. As such, this process enables robust and interpretable reasoning-state identification while preserving coverage of the dominant failure cases.

A.2 Actions (Rule Conclusion)

Table 6 summarizes the four actions considered in this work and their correspondence to representative methods reviewed in Section 2. Each action instantiates an established and empirically effective strategy in prior literature on mathematical reasoning.

Again, these actions are not introduced as novel techniques. Instead, they are treated as modular reasoning strategies that are conditionally activated within the proposed fuzzy reasoning framework. This design avoids reliance on any single strategy and instead integrates diverse reasoning behaviors under a unified fuzzy control mechanism, enabling adaptive action selection conditioned on the diagnosed reasoning states.

B LLM Prompts

This section describes the prompts used in the proposed FLAIR reasoning framework, including those for *Answer Generation*, *Membership Calculation*, *Membership Verification*, and the four reasoning *Actions*.

Notably, the present work does not focus on prompt engineering. The employed prompts are therefore reported for completeness, without additional refinement or optimization. Further performance gains may be achieved by incorporating advanced prompt design or task-specific prompt tuning, which is left for future investigation.

B.1 Prompts for Answer Generation

The following prompt is used to generate solutions to the mathematical problem. The output consists of 4 aspects, including *extracted conditions*, *intermediate calculations*, and a provisional *answer*, where `{question}` denotes the input question content.

At the initial stage, the prompt is executed without any additional action and the solution is generated directly. When a reasoning action is triggered by the fuzzy reasoning rules, the corresponding action prompt is inserted into the `{Optional_Action_Block}` of the prompt, enabling conditional refinement of the reasoning process.

Prompt for Answer Generation

```
Mathematical Problem: {question}

Solve the mathematical problem given
    above, then generate the output in
    the exact JSON format:
{"extracted condition details": ...,
    "intermediate calculation details":
    ..., "answer details": ...}

{Optional_Action_Block}

Output Specifications:
- extracted condition details: Extract
    the numerical values and their
    associated units directly from the
    given problem. Do not infer or
    convert units.
- intermediate calculation details:
    Provide a concise, structured
    calculation process employing
    logical reasoning and mathematical
    operations. All mathematical
    operations and intermediate
    numerical results must be included.
- answer details: Provide the final
```

Table 5: Mapping from our proposed error taxonomy to identified fine-grained error types in (Yin et al., 2025).

Category	Error Type (Yin et al., 2025)
Requirements Error (15.4%)	Misunderstanding of problem requirements Misinterpretation of what the problem is asking for Ambiguous problem parameters Misinterpreted conditions led to incorrect assumptions Incorrect equation setup due to misinterpreted conditions Insufficient understanding or consideration of problem constraints
Reasoning Error (43.6%)	Lack of logical reasoning in arriving at the answer Incorrect application of mathematical formulas or concepts Inconsistent application of formulas Improper application of multiplication relationships Incorrect application of combinatorial principles Misinterpretation of geometric relationships Misinterpretation of scaling language Misapplication of the probability formula Incorrectly assumed equivalence of different algebraic expressions Overreliance on assumptions instead of analysis Overcomplication by introducing irrelevant elements Misapplication of modular reasoning The notation led to confusion in mathematical operations Misplaced focus on solving an unnecessary variable Failure to distinguish between selling price and cost price Lacks thorough analysis of boundary conditions Assumed independence of overlapping events
Calculation Error (17.9%)	Miscalculation during algebraic manipulation Miscalculation during the equation solving process Incorrect calculation during simplification steps Inconsistent variable substitutions Incorrect calculation of the least common multiple (LCM) Incorrect application of trigonometric functions Unit error
Answer Error (15.4%)	Lack of simplification Lack of verification for final answer Failure to consider all possible solutions Misunderstanding of expected answer format Misinterpretation of rounding rules Irrelevant content

```
    numerical result. The value must be
    a numeric value. Do not include
    letters or text.

- Output ONLY the JSON-format result. Do
    not include any other text or
    explanations.
```

B.2 Prompt for Membership Calculation

The following prompt is used to compute fuzzy membership values from the LLM response. The output (in JSON format) consists of four membership scores over predefined fuzzy sets, where `Question: {question}` and `Solution: {extracted condition details} {intermediate calculation details} {answer details}`, respectively, represent the given question and the structured intermediate solution generated in the previous step. The fuzzy set examples included in the membership calculation prompt are generated by perturbation methods as specified in Appendix C.

Prompt for Membership Calculation

```
Your task is to assess the correctness
    of the provided solution of a
```

Table 6: Employed reasoning actions for LLM mathematic problem-solving. Colors indicate the intervention categories to which each action belongs: Prompt Engineering, Reframing, Reasoning Strategies, and Verification (as discussed in Section 2).

Action	Description
($\mathcal{A}_1$) Reasoning Example Provision	Supplying exemplar reasoning chains (Diao et al., 2024; Huang et al., 2024; Yin et al., 2025).
($\mathcal{A}_2$) Entity–Relation Extraction	Extracting entity-level information from the problem statement (Liang et al., 2023; Zhou et al., 2024)
($\mathcal{A}_3$) Chain Decomposition	Decomposing complex reasoning chains (Zhang et al., 2024a; Yang et al., 2025; Leang et al., 2025).
($\mathcal{A}_4$) Error Step Identification	Identifying the first erroneous step and error type (Wu et al., 2025; Y. Li et al., 2025).

```
    mathematical problem, then calculate
    its fuzzy membership degrees for
    each of four fuzzy sets. Each fuzzy
    set defines a different aspect of
    correctness in the solution, and
    each fuzzy membership degree must be
    a float between 0.0 and 1.0, where
    1.0 indicates fully correct and 0.0
    indicates fully wrong.

Four fuzzy sets:
1. Extraction Fidelity degree
   - Definition: Accuracy and precision
    of extracted numeric or textual
    information
2. Operational Accuracy degree
   - Definition: Completeness and
    correctness of operations or applied
    procedures
3. Step Validity degree
   - Definition: Logical soundness and
    correctness of intermediate
    reasoning steps
4. Answer Conformity degree
   - Definition: Adherence to the
    required answer format and
    correctness

Example 1 of fuzzy membership
    calculatioin:
Question: {"A robe takes 2 bolts of blue
    fiber and half that much white
    fiber. How many bolts in total does
    it take?"}
Solution: {"2 bolts, half"}{"- First,
    determine the amount of blue fiber
    needed, which is 2 bolts
- Then, calculate the amount of white
    fiber, which is double the blue
    fiber (Wrong)
- Double of 2 bolts is 2 * 2 = 4 bolts
- Finally, add the amount of blue and
    white fiber to get the total: 2
    bolts (blue) + 4 bolts (white) = 6
    bolts"}{6}
Expected Output: {
"Extraction Fidelity": 1.0,
"Operational Accuracy": 1.0,
"Step Validity": 0.7,
"Answer Conformity": 1.0
}

Example 2 of fuzzy membership
    calculatioin:
Question: {"John drives for 3 hours at a
    speed of 60 mph and then turns
    around because he realizes he forgot
    something very important at home. He
    tries to get home in 4 hours but
    spends the first 2 hours in
    standstill traffic. He spends the
    next half-hour driving at a speed of
    30mph, before being able to drive
    the remaining time of the 4 hours
    going at 80 mph. How far is he from
    home at the end of those 4 hours?"}
Solution: {"3 hours, 75 mph (Wrong), 4
    hours, 2 hours, 30 mph, 95 mph
    (Wrong)"}{"- First, calculate the
    distance John drove before turning
    around: distance = 75 mph * 3 hours
    = 225 miles
- Then, calculate the distance John
    drove in the first 2 hours: 0 miles
    (standstill traffic)
- Next, calculate the distance in the
    next half-hour: 30 mph * 0.5 hours =
    15 miles
- After that, calculate remaining time
    at high speed: 4 - 2 - 0.5 = 1.5
    hours
- Now, calculate the distance at 95 mph:
    95 mph * 1.5 hours = 190 miles
    (Wrong)
- Total distance toward home: 15 + 190 =
    205 miles
- Distance from home: 225 - 205 = 20
    miles"}{20}
Expected Output: {
"Extraction Fidelity": 0.7,
"Operational Accuracy": 0.8,
"Step Validity": 1.0,
"Answer Conformity": 1.0
}

Now calculate the fuzzy membership
    degrees of the following solution of
    a mathematical problem:
```

```
Question: {question}
Solution: {extracted condition
    details}{intermediate calculation
    details}{answer details}

Generate the output in the exact JSON
    format:
{
  "Extraction Fidelity degree": <value>,
  "Operational Accuracy degree": <value>,
  "Step Validity degree": <value>,
  "Answer Conformity degree": <value>
}
- Output ONLY the JSON-format result. Do
    not include any other text or
    explanations.
```

B.3 Prompts for Membership Verification

The following prompt is used to verify fuzzy membership values. Specifically, the LLM is prompted to evaluate the coherence of the assigned membership degrees for *Extraction Fidelity details*, *Operational Accuracy details*, *Step Validity details*, and *Answer Conformity details*. The associated fuzzy set examples within the prompts are also generated with perturbation methods, as specified in Appendix C.

Prompt for Membership Verification

```
Your task is to evaluate or judge
    whether provided fuzzy membership
    degrees accurately reflect solution
    quality. You must evaluate FOUR
    fuzzy sets independently. Below are
    the evaluation criteria and examples
    (*examples are optional).

Fuzzy Set 1: Extraction Fidelity
Criterion: Evaluates whether all
    critical numbers, conditions, and
    the goal from the question are
    correctly identified and extracted.
Example 1:
    - Question: "Josh decides to try
    flipping a house. He buys a house
    for $80,000 and then puts in $50,000
    in repairs. This increased the value
    of the house by 150%. How much
    profit did he make?"
    - Extracted condition: [$120,000
    (Wrong), [repair cost omitted], 150%]
    - The membership degree of
    Extraction Fidelity should be 0.5

Example 2:
    - Question: "James decides to run 3
    sprints 3 times a week. He runs 60
    meters each sprint. How many total
    meters does he run a week?"
    - Extracted condition: [3 sprints, 3
    times a week, 60 meters]
    - The membership degree of
    Extraction Fidelity should be 1.0

Fuzzy Set 2: Operational Accuracy
Criterion: Evaluates the correctness of
    each individual arithmetic operation
    (addition, subtraction,
    multiplication, and division).
Example 1:
    - Question: "James decides to run 3
    sprints 3 times a week. He runs 60
    meters each sprint. How many total
    meters does he run a week?"
    - Calculation Process: "- First,
    calculate the total meters run in
    one set of sprints: 3 sprints * 60
    meters/sprint = 270 meters (Wrong)
    - Then, calculate the total meters
    run in a week: 270 meters/set * 3
    sets/week = 810 meters."
    - The membership degree of
    Operational Accuracy should be 0.8

Example 2:
    - Question: "Janet's ducks lay 16
    eggs per day. She eats three for
    breakfast every morning and bakes
    muffins for her friends every day
    with four. She sells the remainder
    at the farmers' market daily for $2
    per fresh duck egg. How much in
    dollars does she make every day at
    the farmers' market?"
    - Calculation process: "- First,
    calculate the total number of eggs
    laid by the ducks per day: 16 eggs
    - Then, subtract the eggs Janet eats
    for breakfast: 16 - 3 = 13 eggs
    - Next, subtract the eggs used for
    baking muffins: 13 - 4 = 9 eggs
    - Finally, calculate the total
    amount made by selling the remaining
    eggs: 9 eggs * $2/egg = $18"
    - The membership degree of
    Operational Accuracy should be 1.0

Fuzzy Set 3: Step Validity
Criterion: Evaluates whether the logical
    flow and methodological reasoning
    steps are sound.
Example 1:
    - Question: "Every day, Wendi feeds
    each of her chickens three cups of
    mixed chicken feed. She gives the
    chickens their feed in three
    separate meals. In the morning, she
    gives her flock of chickens 15 cups
    of feed. In the afternoon, she gives
    her chickens another 25 cups of
    feed. How many cups of feed does she
    need to give her chickens in the
    final meal of the day if the size of
    Wendi's flock is 20 chickens?"
    - Calculation process: "- First,
    calculate the total amount of feed
    needed for all chickens for the day:
    3 cups/chicken * 20 chickens = 60
    cups
```

```
    - [Step omitted: calculating total
    feed already given]
    - Finally, the remaining feed needed
    is 60 cups - 25 cups = 35 cups"
    - The membership degree of Step
    Validity should be 0.6

Example 2:
    - Question: "Toulouse has twice as
    many sheep as Charleston. Charleston
    has 4 times as many sheep as
    Seattle. How many sheep do Toulouse,
    Charleston, and Seattle have
    together if Seattle has 20 sheep?"
    - Calculation process: "- First,
    determine the number of sheep in
    Seattle, which is 20
    - Then, calculate the number of
    sheep in Charleston, which is 4
    times the number in Seattle: 4 * 20
    = 80
    - Next, find the number of sheep in
    Toulouse, which is twice the number
    in Charleston: 2 * 80 = 160
    - Finally, sum the number of sheep
    in all three locations: 20 + 80 +
    160 = 260"
    - The membership degree of Step
    Validity should be 1.0

Fuzzy Set 4: Answer Conformity
Criterion: Evaluates the final answer's
    correctness regarding the numerical
    value, units, and required format.
Example 1:
    - Question: "Carla is downloading a
    200 GB file. Normally she can
    download 2 GB/minute, but 40% of the
    way through the download, Windows
    forces a restart to install updates,
    which takes 20 minutes. Then Carla
    has to restart the download from the
    beginning. How long does it take to
    download the file?"
    - Calculation process: "- First,
    calculate 40% of 200 GB: 0.4 * 200
    GB = 80 GB
    - Carla downloads 80 GB at 2
    GB/minute, so time taken is 80 GB /
    2 GB/minute = 40 minutes
    - After the restart, Carla has to
    download the entire 200 GB file
    - Time for remaining: 200 GB / 2
    GB/minute = 100 minutes
    - Adding the initial download time,
    the restart time, and the time to
    download: 40 minutes + 100 minutes =
    140 minutes"
    - Answer: 100 (Wrong)
    - The membership degree of Answer
    Conformity should be 0.6

Example 2:
    - Question: "Eliza's rate per hour
    for the first 40 hours she works
    each week is $10. She also receives
    an overtime pay of 1.2 times her
    regular hourly rate. If Eliza worked
    for 45 hours this week, how much are
    her earnings for this week?"
    - Calculation process: "- First,
    calculate the earnings for the first
    40 hours: 40 hours * $10/hour = $400
    - Then, calculate the overtime
    hours: 45 hours - 40 hours = 5 hours
    - Next, calculate the overtime pay
    rate: $10/hour * 1.2 = $12/hour
    - After that, calculate the overtime
    earnings: 5 hours * $12/hour = $60
    - Finally, calculate the total
    earnings: $400 + $60 = $460"
    - Answer: 460
    - The membership degree of Answer
    Conformity should be 1.0

Given the above evaluation criteria and
    examples (*optional), consider the
    following:

Question: {question}
Solution: {extracted condition details}
    {intermediate calculation details}
    {answer details}
Provided membership degrees:
- Extraction Fidelity degree:
    {membership_extraction}
- Operational Accuracy degree:
    {membership_operational}
- Step Validity degree: {membership_step}
- Answer Conformity degree:
    {membership_answer}

Instructions:
For each fuzzy set:
1. Re-evaluate the provided membership
    degree based on the above evaluation
    criteria and examples (*optional).
2. If the absolute difference between
    the reassessed membership and the
    provided membership is less than
    0.3, mark the provided membership as
    supported; otherwise, mark it as not
    supported. Provide a concise
    justification.

Return a JSON object with FOUR entries:
{
  "Extraction Fidelity": {
    "supported": true or false,
    "reason": "Concise justification"
  },
  "Operational Accuracy": {
    "supported": true or false,
    "reason": "Concise justification"
  },
  "Step Validity": {
    "supported": true or false,
    "reason": "Concise justification"
  },
  "Answer Conformity": {
    "supported": true or false,
    "reason": "Concise justification"
  }
}
```

```
- Output ONLY the JSON-format result. Do
    not include any other text or
    explanations.
```

B.4 Prompts for Actions

This subsection provides prompts corresponding to four reasoning actions employed in the framework. Given `Question: {question}` and the structured intermediate solution `Solution: {extracted condition details} {intermediate calculation details} {answer details}`, each action prompt guides the model to perform a targeted reasoning adjustment. Importantly, actions are conditionally triggered according to the diagnosed reasoning states (or its associated fuzzy membership values), allowing multiple actions to be applicable under different membership configurations.

To begin with, the following prompt is used for the *Reasoning Example Provision* action, which aims to provide correct reasoning exemplars. The exemplars can be obtained directly from the training dataset (such as GSM8K) and are used to reinforce valid reasoning structures.

Prompt for Action: Reasoning Example Provision

```
The following exemplar demonstrates a
    valid reasoning chain for a related
    mathematical problem.

Example of Reasoning Chain:
{
    "question": "Every day, Wendi feeds
    each of her chickens three cups of
    mixed chicken feed, containing
    seeds, mealworms and vegetables to
    help keep them healthy. She gives
    the chickens their feed in three
    separate meals. In the morning, she
    gives her flock of chickens 15 cups
    of feed. In the afternoon, she gives
    her chickens another 25 cups of
    feed. How many cups of feed does she
    need to give her chickens in the
    final meal of the day if the size of
    Wendi's flock is 20 chickens?",
    "extracted condition details": "20
    chickens; 3 cups per chicken; 15
    cups; 25 cups",
    "intermediate calculation details":
    "Total feed needed = 20 * 3 = 60
    cups; Feed given in morning and
    afternoon = 15 + 25 = 40 cups; Feed
    needed in final meal = 60 - 40 = 20
    cups",
    "answer details": "20"
}

Using the provided exemplars solely as
    guidance to logically derive the
    solution to the given mathematical
    problem.
```

The next prompt is used for the *Entity–Relation Extraction* action, which directs the model to explicitly extract key entities, quantities, and relations from the problem statement. This action promotes structured interpretation of the input and reduces ambiguity in the subsequent reasoning process.

Prompt for Action: Entity–Relation Extraction

```
As an auxiliary reasoning step,
(1) Extract entities and relationships
    from the given problem.
(2) Use the extracted relationships to
    logically derive the answer.
This step is for internal reasoning only
    and MUST NOT affect the output
    format.

Reasoning guidance:
1. Extract Entities: Identify all
    relevant entities in the question.
    Classify each entity into one of the
    following types: object, person,
    number, action, attribute. Assign a
    unique identifier to each entity
    (e.g., E1, E2). Format: [Entity ID]:
    [Entity Name] (type).

2. Explicit Relationship Extraction:
    Identify all explicitly stated
    relationships between entities based
    on context. Supported relationship
    types include: ownership,
    comparison, action effect, part-of.
    Format each relationship as a
    triple: (EA, EB, Relation).

3. Relationship Confidence Evaluation:
    Assign a confidence score to each
    explicit relationship. The score
    ranges from 0.0 (invalid) to 1.0
    (certain). Format: (EA, EB,
    Relation, Score).

4. Implicit Relationship Inference:
    Infer Infer unstated relationships
    using deductive reasoning, such as:
    temporal sequence, logical
    dependency, causal chain. Format
    each implicit relationship as a
    triple: (EA, EB, Relation).

Using all extracted entities and
    relationships (both explicit and
```

```
    implicit), logically derive the
    solution to the given mathematical
    problem. Do NOT reveal any entities,
    relationships, or intermediate
    reasoning scores in the final answer.
```

The third prompt is used for the *Problem Decomposition* action, which restructures the reasoning process into explicit and logically ordered substeps. This action aims to reduce failure modes associated with implicit or entangled reasoning trajectories.

Prompt for Action: Problem Decomposition

```
As an auxiliary reasoning step,
    decompose the given problem into a
    sequence of logically coherent
    sub-problems. Each sub-problem
    should represent a necessary
    reasoning step toward the final
    solution and be solvable
    independently.

For each sub-problem:
1. Clearly state the sub-problem.
2. Provide a concise and correct
    solution.
3. Ensure that the progression between
    sub-problems is logically consistent.

Then, use the reasoning implied by the
    sub-problems to derive the solution
    to the original question.
```

At last, the prompt for the *Incorrect Step Identification* action is provided hereafter, which guides the model to identify reasoning steps that are inconsistent, missing, or incorrect. This action takes the original question and the intermediate solution as input to localize reasoning flaws, enabling targeted refinement of the reasoning process.

Prompt for Action: Incorrect Step Identification

```
As an auxiliary reasoning step, verify
    the correctness of the provided
    solution with respect to the
    original question.

Inputs:
- Question: {question}
- Solution: {extracted condition
    details} {intermediate calculation
    details} {answer details}
```

```
Task:
1) Determine whether the provided
    solution is correct.
2) If incorrect, locate the first
    erroneous step and assign exactly
    one error type from the list below.
3) Based on the identified issue,
    internally derive a corrected
    solution.

Error type definitions:
- referencing context value error:
    incorrectly referencing numerical
    values from the original question.
- referencing previous step value error:
    incorrectly referencing results from
    a previous step.
- unit conversion error: incorrect unit
    conversion affecting operands.
- operator error: incorrect operator
    used in a computation.
- calculation error: operands/operators
    are correct, but the numerical
    result is incorrect.
- missing step: a required step is
    omitted; adding it corrects the
    solution.
- confusing formula error: an incorrect
    formula is applied.
- adding irrelevant information:
    information not present in the
    question is introduced and affects
    the result.


Using identified issues to logically
    derive the corrected solution to the
    given mathematical problem. All
    error analysis must remain hidden;
    only the final answer to the
    original question should be provided.
```

C Annotated Fuzzy Set samples

Classical fuzzy theory relies on explicitly defined membership functions (*e.g.*, triangular, Gaussian, or bell-shaped functions) (Zadeh, 1965), where a numerical input is mapped to a membership degree through a fixed functional form. In contrast, the present FLAIR framework operates in the context of LLMs, for which predefined analytical membership functions are neither natural (textual input) nor expressive enough to capture reasoning-level uncertainty. Instead, fuzzy membership is inferred through LLM-driven evaluation of reasoning behavior. A key challenge arising from this design is how to reliably guide LLMs to produce consistent and meaningful membership assessments.

To address this issue, this section introduces *annotated fuzzy set samples*, which serve as grounded reference examples for membership estimation.

Similar to traditional fuzzy theory, **each fuzzy set sample consists of two components**: an input instance (x) and an associated membership degree (μ) indicating the extent to which the instance belongs to a given fuzzy set. Obviously, these samples provide supervision signals that can be leveraged for *membership calculation or verification*.

The core idea is to construct such samples by introducing **controllable perturbations** to existing training instances. Many mathematical reasoning datasets, such as `GSM8K`, provide not only correct answers but also explicit step-by-step reasoning processes. These reasoning traces offer a natural foundation for generating fuzzy set samples by selectively modifying specific aspects of the reasoning process while keeping others intact. Importantly, the proposed perturbation strategy is not tied to any particular dataset, as the same construction procedure can be applied whenever annotations are available, either obtained naively or through manual labeling.

Specifically, controllable perturbations are introduced to systematically alter reasoning steps, numerical calculations, or logical dependencies, thereby producing instances that *exhibit varying degrees of deviation from correct reasoning*. Accordingly, each perturbed instance x is then annotated with a corresponding membership degree μ that reflects *its alignment with a particular fuzzy set*.

Recall that Section 3.1 introduces four predefined fuzzy sets. Next, we introduce perturbation strategies used to construct annotated fuzzy samples with both degraded ($\mu < 1$) and full ($\mu = 1$) membership values.

Extraction Fidelity (EF) Perturbations. The *Extraction Fidelity* fuzzy set models the extent to which a solution accurately extracts and utilizes all necessary numerical conditions from the problem statement.

Assume that one problem instance contains n_{cond} numerical conditions. To generate perturbed samples (x, μ_{EF}), a subset of these conditions is randomly corrupted or removed. Specifically, perturbations are applied by either (1) *replacing* the original numerical values with incorrect ones by adding a random offset $\Delta_i \in [1, 20]$, or (2) *omitting* the corresponding conditions entirely from the extracted representation. To model varying degrees of deviation from correct reasoning, each perturbed condition i is associated with a severity score $\delta_i \in (0, 1]$, where omission is treated as the most severe perturbation and value replacement incurs a severity proportional to the relative offset magnitude. Concretely, the severity score is defined as $\delta_i = 1$ for omitted conditions and $\delta_i = \min(1, |\Delta_i|/20)$ for replaced values, while unperturbed conditions have $\delta_i = 0$. The Extraction Fidelity membership degree is then defined as

$$\mu_{\text{EF}} = 1 - \frac{1}{n_{\text{cond}}} \sum_{i=1}^{n_{\text{cond}}} \delta_i, \tag{5}$$

which yields the fuzzy membership score that decreases smoothly with both the number and the severity of perturbations.

Operational Accuracy (OA) Perturbations. The *Operational Accuracy* fuzzy set models the extent to which mathematical operations and their resulting computations are performed correctly.

Assume that one solution contains n_{eq} computational equations. To construct (x, μ_{OA}), similar to the Extraction Fidelity perturbations, for each equation j ($j \in [1, n_{\text{eq}}]$), the computed numerical result can be altered by multiplying it with a factor $\alpha_j \in [1.5, 2.5]$ while keeping the original operation unchanged. Concretely, the severity score is defined as $\delta_j = \min(1, |\alpha_j - 1|/1.5)$, while unperturbed equations have $\delta_j = 0$. Similar to μ_{EF}, the Operational Accuracy membership degree is then defined as $\mu_{\text{OA}} = 1 - \frac{1}{n_{\text{eq}}} \sum_{j=1}^{n_{\text{eq}}} \delta_j$.

Steps Validity (SV) Perturbations. The *Steps Validity* fuzzy set models the extent to which a reasoning process is logically coherent, complete, and semantically well-formed across successive steps.

Assume that a solution contains n_{step} reasoning steps. To construct perturbed samples (x, μ_{SV}), the following perturbation strategies are considered: (1) *step omission*, where an essential intermediate reasoning step is removed, resulting in an incomplete reasoning chain, (2) *operator substitution*, where arithmetic operators within a reasoning step are replaced with incorrect alternatives, breaking logical consistency, and (3) *semantic substitution*, where reasoning expressions are replaced with semantically mismatched but syntactically plausible alternatives (e.g., replacing "the total amount" with "the average amount", or substituting "more" with "few").

The severity score $\delta_j \in [0, 1]$ is manually defined according to the perturbation type applied to

step j as

$$\delta_j = \begin{cases} 1.0, & \text{if step } j \text{ is omitted,} \\ \gamma_{\text{op}}, & \text{operator substitution,} \\ \gamma_{\text{sem}}, & \text{semantic substitution,} \\ 0, & \text{if step } j \text{ is unperturbed,} \end{cases} \quad (6)$$

where $\gamma_{\text{op}} \in (0.6, 0.8]$ denotes medium-to-high severity induced by incorrect operator usage, and $\gamma_{\text{sem}} \in (0.3, 0.5]$ denotes lower-to-medium severity caused by semantic mismatch. The Steps Validity membership degree is then defined as $\mu_{\text{SV}} = 1 - \frac{1}{n_{\text{step}}} \sum_{j=1}^{n_{\text{step}}} \delta_j$.

Answer Conformity (AC) Perturbations. The *Answer Conformity* fuzzy set models the extent to which the final answer conforms to the expected semantic constraints and output format specified by the problem.

Assume that a solution yields a single final answer. To construct perturbed samples (x, μ_{AC}), three answer-level perturbation strategies are considered with equal weight: (1) *unit change*, where the answer unit is replaced with a semantically incorrect unit; (2) *format change*, where the answer format is altered, such as providing a non-integer value when an integer count is required (e.g., "number of apples"), or violating structured formats (e.g., time expressed in `hh:mm`); and (3) *representation inconsistency*, where extraneous symbols or textual descriptors are added despite the requirement for a pure numeric answer.

Each perturbation type contributes an equal penalty of $1/3$ to answer conformity. For each instance, a random subset of these perturbations is selected and applied. The fuzzy membership degree is then defined as

$$\mu_{\text{AC}} = 1 - \frac{1}{3} \sum_{i=1}^{3} \mathbb{I}\big[\text{perturbation } i \text{ is applied}\big], \quad (7)$$

where $\mathbb{I}[\cdot]$ denotes the indicator function.

Meaning-Preserving Perturbations. The perturbation strategies described above deliberately introduce controlled deviations to transform correct instances into samples with $\mu_{\text{EF}}, \mu_{\text{OA}}, \mu_{\text{SV}}, \mu_{\text{AC}} < 1$. In addition, perturbations are also employed to generate samples with full membership ($\mu = 1$) without altering the original semantic or reasoning correctness.

These meaning-preserving perturbations fall into two categories. For *textual* components, synonym substitution [5] is applied to replace words or phrases with semantically equivalent alternatives. For *numerical* components, a numerical value is rewritten as an alternative expression with identical semantics (*e.g.*, rewriting 17 as $10 + 7$ or seventeen). Such perturbations preserve all reasoning-relevant information and do not affect the correctness of extraction, computation, reasoning steps, or the final answer.

Overall, by incorporating both degrading and meaning-preserving perturbations, the constructed samples cover the full range of fuzzy membership values. These fuzzy set samples are then used for membership calculation or verification, and reasoning exemplars (if required).

D Base LLMs

In the main experiments, the same base LLM is used for answer generation, fuzzy membership calculation, and membership verification. Despite the simplicity, it may introduce implicit biases in reasoning-state assessment. Specifically, when the same model performs both generation and verification, reasoning errors produced during generation may be partially overlooked during evaluation, as the verifier could share similar inductive biases and internal representations. This concern is consistent with prior observations in LLM self-evaluation and self-consistency studies, which report that models tend to under-detect their own reasoning flaws (Xu et al., 2024b; Spiliopoulou et al., 2025).

To mitigate this issue, a decoupled setting is adopted, in which one model performs solution generation, while a separate model is used exclusively for membership calculation/verification. Specifically, Llama3.3-70B and GPT-4 are considered as base models, while the other configuration remains unchanged.

As observed from Table 8, when GPT-4 is used for generation and Llama3.3-70B is used for fuzzy, performance remains largely comparable, yet with marginal accuracy degradation. This indicates that a stronger generator tends to produce reasoning outputs that are robust to variations in the (weaker) verifier. In contrast, when Llama3.3-70B is used for generation and GPT-4 is used for verification, accuracy is consistently higher than when Llama3.3-

[5] Implementation from https://github.com/dsfsi/textaugment.

Table 7: Comparison of computational complexity and model accuracy using GPT-4.

Dataset	Method	Acc.	Avg API Calls	Avg Tokens	Avg Tok./API
GSM8K	Self-Refine (Madaan et al., 2023)	94.5	4.1	4515.0	1101
	LBS3 (Luo et al., 2025b)	94.9	5.0	2611.0	522
	SAFE (Liu et al., 2025a)	96.0	19.8	10812.0	546
	FLAIR	**98.4**	**3.5**	**477.4**	**136**
MATH	Self-Refine (Madaan et al., 2023)	60.5	7.7	6823.0	886
	LBS3 (Luo et al., 2025b)	64.2	6.0	3577.0	596
	SAFE (Liu et al., 2025a)	80.4	34.7	23009.0	663
	FLAIR	**82.2**	**6.1**	**917.5**	**150**

Solution	Fuzzy	GSM8K	SVAMP
GPT-4	GPT-4	98.4	97.0
GPT-4	Llama3	96.4	96.3
Llama3	GPT-4	96.9	96.7
Llama3	Llama3	96.2	96.5

Table 8: Decoupled LLMs for **solution** generation, and for **fuzzy** membership calculation/verification.

70B is used for both generation and verification. This gap suggests that a stronger verifier is more sensitive to reasoning errors produced by a comparatively weaker generator, leading to more effective verification and improved overall performance.

E Alternative Membership Calculators

In the main experiments, fuzzy memberships are estimated using prompt-based calculators that include both fuzzy set definitions and illustrative examples (see Appendix B.2). For comparison, three alternative variants are evaluated: (i) **DefOnly**, which uses prompts containing only fuzzy set definitions; (ii) **ExOnly**, which relies solely on illustrative examples; and (iii) **SmallLM**, which employs a separately trained lightweight model for direct membership estimation.

For **DefOnly** and **ExOnly**, the variants are implemented by simply removing the corresponding components (illustrative examples or fuzzy set definitions) from the calculation prompt, while keeping all other prompt content unchanged. For **SmallLM**, the `DeBERTa-v3-base` model (He et al., 2023) is adopted as the backbone and trained to directly predict fuzzy membership scores. The input is constructed as:

```
[CLS] Question [EF] EF statement [OA]
OA statement [SV] SV statement [AC] AC
statement [SEP]
```

The latent representations corresponding to `[EF/OA/SV/AC]` are fed into four evaluation heads. Each head consists of a one-hidden-layer feed-forward network, with the sigmoid function as the activation. The model is optimized using AdamW with a learning rate of 2×10^{-5}, a warmup ratio of 5%, gradient clipping at 1.0, a batch size of 8, and a maximum input length of 512 tokens.

F Detailed Computational Efficiency Results

Detailed efficiency comparisons under GPT-4, including average API calls, total token usage, and tokens per API call on GSM8K and MATH, are provided in Table 7. In addition, we conduct an empirical comparison via official codes released by three prior methods (ie., Self-Refine, LBS3 and SAFE), and the results on GSM8K and MATH (using GPT-4 as the backbone) are summarized in Table 7. As shown, `FLAIR` consistently achieves the lowest average computational cost among all compared methods (with the lowest Avg Tokens / API). This suggests that the improvements are not attributable to increased test-time computation, but rather to the proposed fuzzy uncertainty modelling that enables more targeted and efficient reasoning.

G Details of baselines

In Section 4.2, we compare the proposed `FLAIR` with 17 baselines. Their details are shown in Table 9, based on their order in the main experiments.

Table 9: Details of employed baselines in this study.

Method	Description
FOBAR (Jiang et al., 2024a)	FOBAR uses forward reasoning to generate diverse candidate solutions and backward reasoning to verify candidates.
LBS3 (Luo et al., 2025b)	LBS3 recalls easy-to-hard proxy queries from a curated query pool and adaptively uses their solutions as exemplars to guide LLM reasoning.
Self-Contrast (Zhang et al., 2024b)	Self-Contrast generates solutions from diverse perspectives, contrasts their differences, and summarizes inconsistencies to guide self-correction.
WizardMath (Luo et al., 2025a)	WizardMath applies Reinforcement Learning with process-level signals generated via instruction evolution to improve mathematical reasoning.
MetaMath (Yu et al., 2024)	MetaMath bootstraps rewritten problem variants to finetune mathematical reasoning models.
MathFusion (Pei et al., 2025)	MathFusion synthesizes existing problems via sequential, parallel, and conditional fusion to finetune mathematical reasoning models.
SAFE (Liu et al., 2025a)	SAFE translates each reasoning step into Lean 4 formal statements to detect hallucinations in mathematical reasoning.
DART (Tong et al., 2024)	DART employs difficulty-aware rejection to construct compact, high-quality datasets for finetuning mathematical reasoning models.
MMIQC (Liu et al., 2025b)	MMIQC iteratively composes new questions from seed problems using LLMs to generate high-quality finetuning data.
KP (Huang et al., 2025)	KP focuses on key points and exemplar practices from authentic data to produce large-scale finetuning datasets.
StepCo (Wu et al., 2025)	StepCo iteratively verifies and revises generated solution steps to correct errors and reduce token usage.
EAP (Yin et al., 2025)	EAP categorizes potential reasoning errors and incorporates relevant examples into prompts to guide mathematical reasoning.
MultiTAG (Yao and Yadav, 2025)	MultiTAG invokes multiple tools at each reasoning step and aggregates their outputs via majority voting to improve accuracy.
SKTO (Lin et al., 2025)	SKTO provides binary feedback on intermediate steps and final answers to guide reasoning revision.
MCoT (Yang et al., 2025)	MCoT compresses prior steps into question preconditions and applies code-assisted self-correction to simplify reasoning.
AceMath (Liu et al., 2025c)	AceMath applies supervised finetuning on curated synthetic prompts and solutions to improve mathematical reasoning.
SGR (Cao et al., 2025)	SGR encourages reflection on fine-grained stepwise decisions during inference to guide mathematical reasoning.