PRISM: Refracting Hallucinations in Chain-of-Thought Reasoning via Multi-Dimensional Verification

Anonymous ACL submission

Abstract

In recent years, Chain-of-Thought (CoT) verification has emerged as a critical research direction. However, existing approaches largely focus on the quality of intermediate reasoning or final answer correctness, while hallucinations arising from the initial stage of question understanding remain underexplored. To address this gap, we propose a unified framework—PRISM (**P**rogressive **R**easoning with **I**nstructional and **S**trategic **M**ulti-dimensional Verification) that jointly tackles all three aspects. We introduce a Commonsense-Augmented Progressive Instructional Reasoning (CPIR) method, designed to alleviate condition hallucination while utilizing commonsense to capture relevance between conditions and questions. Then we develop Multi-Dimensional Heterogeneous Collaborative Verification (MHCV), which strategically validates reasoning chains from multiple perspectives to enhance intermediate reasoning quality and question comprehension, thereby mitigating different types of hallucinations. In addition, we propose a Discard-Weighted Voting mechanism to overcome the limitations of traditional voting methods in multi-dimensional verification. Experimental results demonstrate that PRISM consistently improves verification accuracy across conditions, logical reasoning, and question comprehension, yielding more reliable reasoning chains and higher final-answer accuracy compared to strong CoT baselines.

1 Introduction

Step-by-step reasoning is a fundamental capability in both analytical thinking and problem-solving. As early as ancient Greece, philosophers like Socrates emphasized guarding against implicit fallacies through rigorous questioning step by step. In recent years, CoT prompting has advanced the reasoning capability of LLMs to a new level (Wei et al., 2022; Cheng et al., 2024; Zhang et al., 2024), significantly enhancing the effectiveness of reasoning chains in addressing complex problems across a wide range of domains (Li et al., 2025; Mitra et al., 2024; Hu et al., 2025; Nong et al., 2024; Liao et al., 2025; Sanwal, 2025).

Although CoT significantly improves LLM reasoning, its reliability remains inconsistent. As observed by Arcuschin et al. (Arcuschin et al., 2025), once an error occurs, subsequent steps may propagate flawed premises, leading to hallucinations (Chu et al., 2024; Zhao et al., 2023; Feng et al., 2023). This has led to growing interest in verifying CoT-generated reasoning. Verification methods are broadly divided into forward and backward approaches. Forward verification follows the original reasoning flow, validating steps incrementally or holistically (Chu et al., 2024; Weng et al., 2022; Zhu et al., 2024). However, as CoT chains grow longer, it tends to reproduce the original reasoning path (Zhang and Xiong, 2025), making it hard to detect question misinterpretation. Backward verification starts from the final answer to infer original premises (Jiang et al., 2024), mimicking human reverse reasoning. Though more robust for complex tasks, it often overlooks intermediate reasoning steps and tends to reflect only a partial understanding of the question (Weng et al., 2022; Ranaldi and Pucci, 2025; Deb et al., 2024). Therefore, existing methods struggle to balance the question comprehension, intermediate reasoning quality and final answers' accuracy.

To address these challenges, we propose **Commonsense-Augmented Progressive Instruction Reasoning (CPIR)** in the reasoning stage, which mitigates condition hallucination by extracting key conditions and generating progressive instructional lists, while leveraging commonsense to capture and make explicit the relational links between conditions and the target question. For verification, inspired by ancient Greek step-by-step questioning, we introduce the **Multi-Dimensional Het-**

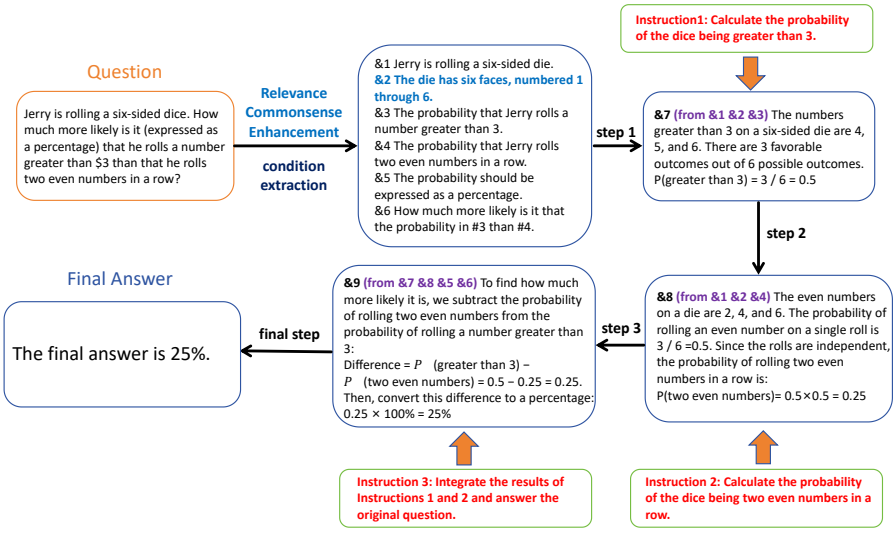

Figure 1: An example of Commonsense-Augmented Progressive Instruction Reasoning

erogeneous Collaborative Verification (MHCV) method. Forward verification detects conditional and logical inconsistencies, while backward verification infers the original question from progressive instructions to identify question misunderstandings. Unlike previous studies, we address hallucination issues from both forward and backward directions, with particular emphasis on question comprehension hallucination—where the LLM misunderstands or forgets the true meaning of the original question during reasoning—a challenge that has received limited attention in prior work.

We present a unified pipeline that jointly operationalizes forward and backward verification with a dedicated module for question-comprehension hallucination. Compared with deductive reasoning approaches (Ling et al., 2024; Zhu et al., 2024), backward verification that reconstruct key conditions in questions (e.g., Self-Verification (Weng et al., 2022)), and tool-based or retrieval-based approaches (e.g., Verify-and-Edit (Zhao et al., 2023)), our framework differs in:

1. We demonstrate that LLMs can inherently generate the commonsense needed to capture the relations linking the given conditions to the target question. By making such relations explicit and introducing progressive instructions, our method mitigates implicit errors and enables structured reasoning to support effective verification.

2. We propose a unified multi-dimensional hallucination verification framework that integrates both forward and backward verification. This approach effectively addresses condition hallucination, logical errors, and question misinterpretations, balancing question understanding, intermediate reasoning quality, and final-answer accuracy.

3. We introduce a *Weighted Discard Voting* strategy that selectively retains partially correct reasoning instead of discarding entire chains for minor issues, thereby enhancing the reliability of final answers.

2 Related Work

Chain-of-Thought Prompting CoT prompting has significantly improved the reasoning ability and interpretability of LLMs in question answering tasks (Wei et al., 2022; Chu et al., 2024; Li et al., 2025; Cheng et al., 2024; Sanwal, 2025). Several studies (Zhou et al., 2022; Lyu et al., 2023; Jung et al., 2022; Qian et al., 2022) proposed a question translator that decomposes a complex query into simpler sub-questions. Similarly, the condition translator (Zhu et al., 2024) breaks down conditional logic for easier downstream reasoning. However, these decomposition strategies often lack interpretability and offer limited guarantees on the correctness of the reformulated components (Huang et al., 2025; Kumar et al., 2025). To address these issues, our method incorporates

commonsense-augmented conditional decomposition and progressive instruction generation.

Verification of Chain-of-Thought CoT verification can be broadly classified into forward and backward methods. Forward approaches rely on either external tools or LLMs. Tool-based methods (Inaba et al., 2023; Wu et al., 2025; Jiang et al., 2025) use code interpreters, calculators, or search engines, while retrieval-based ones (Zhang et al., 2023; Ni et al., 2025; Chen et al., 2025; Zhang et al., 2025) check factual validity. However, these methods face issues in robustness and security. LLM-based strategies rely on the model's own reasoning capabilities, using self-iteration, self-feedback, or stepwise validation (Madaan et al., 2024; Paul et al., 2023; Ling et al., 2024). Despite the great progress, forward verification still struggles to detect implicit fallacies and often fails to capture misunderstandings arising from question comprehension. Backward verification simulates human reverse-checking by masking key information and checking for consistency in inferred reasoning (Jiang et al., 2024; Weng et al., 2022). While this provides a complementary perspective, it is computationally expensive and often overlooks holistic reasoning quality. Furthermore, because it conditions on the final answer, backward verification often reflects only a partial understanding of the question—reconstructing premises that are compatible with the answer but not faithful to the original wording or constraints, and thus overlooking implicit requirements and step-level inconsistencies (Ranaldi and Pucci, 2025; Deb et al., 2024). To overcome these limitations, we propose a multi-dimensional verification framework to more comprehensively address various hallucinations.

Voting Strategic Majority voting is widely used to select the final answer but may not generalize well across different tasks. Some approaches (Wang et al., 2025; Zelikman et al., 2022; Miao et al., 2023) score reasoning paths via auxiliary models to improve quality, at the cost of increased computational load. Others (Zhu et al., 2024; Vacareanu et al., 2024) apply multi-dimensional filtering before voting, but over-filtering can discard valid chains with minor flaws, reducing reliability. To address this, we propose a Discard-Weighted Voting strategy, which uses verification feedback to assign discard scores and is more suitable for multi-dimensional verification screening.

3 The Proposed Method

In this section, we present the PRISM framework for reasoning and verification in Chain-of-Thought. In the reasoning stage, a Condition Display Module leverages commonsense to make conditions and condition–question relations explicit, while an Instruction Generator produces progressive instruction lists tailored to the given question. In the verification stage, we introduce MHCV, a multi-dimensional framework that verifies hallucinations across three complementary aspects. Finally, Discard-Weighted Voting is proposed to select the most reliable candidate answer.

All modules in PRISM are implemented using a single LLM. Different functional roles (e.g., reasoning generator, condition verifier, logic verifier, and backward verifier) are instantiated through role-specific prompts, rather than separate models. The modules operate sequentially within a unified prompting pipeline. Prompts for each stage are detailed in Appendix A.18.

3.1 Commonsense-Augmented Progressive Instruction Reasoning

Prior methods decompose complex problems into sub-information through a translator (Zhu et al., 2024; Wu et al., 2024; Ling et al., 2024; Zhou et al., 2022), thereby simplifying the reasoning process. As illustrated in Figure 1, the *Condition Display Module* extracts the necessary conditions and integrates only the commonsense knowledge that links the conditions to the question, while discarding any irrelevant ones. Then, the *Instruction Generator* produces progressive instructions for intermediate steps, ensuring that each step moves in the correct direction.

We define a complete CoT reasoning process as a tuple (Q, C, R, A), where:

- Q denotes the question, consisting of the final query statement and its associated conditions.
- C denotes the commonsense knowledge linking conditions to the question;
- R denotes the intermediate reasoning process;
- A denotes the final answer derived from the reasoning.

Inspired by Deductive Verification of Chain-of-Thought Reasoning (Ling et al., 2024), we eliminate redundant information to minimize potential

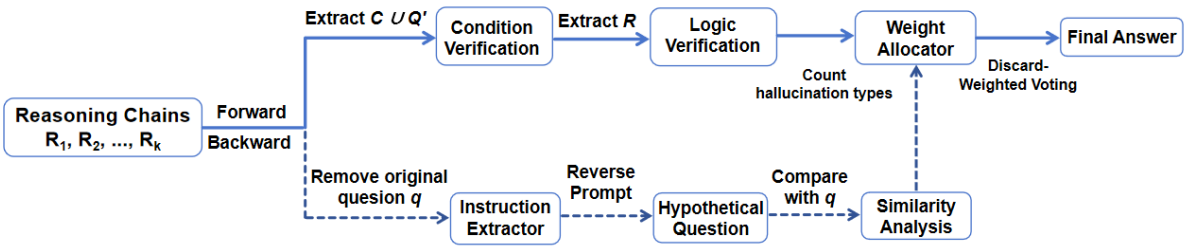

Figure 2: Architecture of Multi-Dimensional Heterogeneous Collaborative Verification

interference with the reasoning. Specifically, the question Q is decomposed into:

$$Q' = \{b_1, b_2, \ldots, b_m, q\}$$

where $\{b_1, b_2, \ldots, b_m\}$ are the minimal background conditions required to answer the question, and q is the final query statement.

Subsequently, the final inquiry q is enriched with relevant commonsense knowledge, resulting in a set. Our method leverages LLM's own capabilities to supplement commonsense without relying on external interfaces, thereby improving both efficiency and security (see Appendix A.10 for details):

$$C = \{c_1, c_2, \ldots, c_p \mid q\}$$

In cases where no supplementary commonsense is deemed necessary, the model outputs "No supplementary commonsense is needed", and C is treated as an empty set. The LLM then performs reasoning based on the combined input $C \cup Q'$.

During the reasoning process, an *Instruction Generator* assigns a progressive instruction to each intermediate reasoning step. This yields a sequence of intermediate reasoning steps:

$$R = \{(i_1, g_1, r_1), (i_2, g_2, r_2), \ldots, (i_n, g_n, r_n)\}$$

where i_n is the n-th progressive instruction, g_n is the index set of conditions required for the n-th step, and r_n represents the content of the n-th intermediate reasoning step.

3.2 Multi-dimensional Heterogeneous Collaborative Verification Framework

Unlike traditional methods that rely solely on forward or backward verification, the MHCV framework combines both to leverage their strengths, as shown in Figure 2. Forward verification detects condition hallucination and logical errors in intermediate steps, while backward verification identifies issues in question comprehension. Beyond final-answer selection, MHCV focuses on enhancing the overall quality of the reasoning chain by ensuring logical coherence and reinforcing the model's comprehension of the question.

3.2.1 Forward Conditional and Logic Verification Method

Forward verification follows the direction of CoT reasoning to examine various types of errors and hallucinations in intermediate reasoning steps. This method consists of two stages:

In the first stage, given the set of intermediate reasoning steps $R = \{(i_1, g_1, r_1), (i_2, g_2, r_2), \ldots, (i_n, g_n, r_n)\}$, the progressive instructions i_k $(1 \leq k \leq n)$ are removed, yielding a simplified reasoning set $R' = \{(g_1, r_1), (g_2, r_2), \ldots, (g_n, r_n)\}$. Subsequently, the decomposed question set Q' is used to extract the foundational background conditions $Q' - q = \{b_1, b_2, \ldots, b_m\}$, which, combined with C, form the complete condition set $(Q' - q) \cup C$. Each intermediate reasoning step (g_k, r_k) $(1 \leq k \leq n)$ is verified against this condition set, which is progressively augmented with any newly inferred conditions.

The verification outcome for the k-th intermediate reasoning step is as follows:

$$\mathcal{V}(g_k, r_k \mid (Q'-q) \cup C) = \begin{cases} \text{true}, & \text{if conditions are met} \\ \text{false}, & \text{otherwise} \end{cases}$$

In the second stage, logical reasoning verification assesses whether each intermediate reasoning step maintains logical coherence. This verification consists of two types: for mathematical reasoning tasks, it detects arithmetic or logical errors within candidate chains; for linguistic reasoning tasks, it identifies semantic inconsistencies. Both types of verification are performed via few-shot prompting. Let $L_k(r_k)$ $(1 \leq k \leq n)$ denote the verification result of the k-th reasoning step, which represents the forward verification outcome for step k.

$$F_k = \mathcal{V}(g_k, r_k \mid (Q' - q) \cup C) \wedge L_k(r_k)$$

The entire candidate chain passes forward verification if and only if all intermediate reasoning steps individually pass their respective forward verification.

Unlike prior works (Weng et al., 2022; Zhao et al., 2023; Ling et al., 2024; Kumar et al., 2025) that decompose problems through external translators or treat reasoning holistically, our approach introduces three key innovations: (i) implicit information is systematically made explicit through a Condition Display Module, (ii) commonsense is directly supplemented by the LLM to capture condition–question relations without relying on external resources, and (iii) progressive instructions are generated to guide reasoning in a controllable, step-by-step manner.

3.2.2 Reverse Instructional Question Backtracking Verification Method

Although verification can ensure conditions, commonsense, and logical coherence, it cannot prevent errors from an initial misinterpretation of the question, arising from semantic ambiguity or gradual drift in reasoning. Such hallucinations are often missed by current verification methods.

To address this issue, we propose **Reverse Instructional Question Backtracking Verification**. Unlike backward methods that merely compare final answers, our approach abstracts away conditions and reasoning details, focusing instead on the instructional trajectory. By extracting intermediate instructions and using them to reconstruct the predicted question, we evaluate semantic alignment with the original question, thereby mitigating question-comprehension hallucination[1] in CoT reasoning.

To verify question consistency, we reconstruct the original question from the instructional flow extracted from sampled reasoning chains. The verification process is detailed in **Algorithm 1**.

The purpose of Reverse Instructional Question Backtracking Verification is to mitigate question-comprehension hallucination. Unlike approaches that evaluate only final answer correctness, our method examines whether the reasoning trajectory aligns with the original intent. By abstracting away numerical values and detailed steps, it enhances verification efficiency and reduces deviations from the intended question. The algorithmic details and reliability analysis of the reverse verification process are provided in Appendices A.5 and A.1.

[1]For a detailed explanation and examples of question-comprehension hallucination, see Appendix A.9.

3.2.3 Discard-Weighted Voting

Most CoT reasoning methods adopt *Majority Voting* to select the most frequent candidate answer, while others (Wang et al., 2025; Zhu et al., 2024) employ auxiliary models or multi-dimensional filtering to refine the outputs. However, in multi-dimensional verification, these strategies face two key limitations: (1) stricter screening often filters out more candidate chains, reducing the overall sample size and weakening the reliability of voting; (2) chains with minor deviations—such as misreferenced conditions or slight phrasal variations—can still produce correct answers, yet Majority Voting may discard them entirely.

To address these issues, we propose **Discard-Weighted Voting** strategy, which assigns different weights to various types of hallucinations in order to compute a discard score for each reasoning chain. A higher score indicates more severe hallucinations and lower reliability. Given k candidate chains, we count the frequency of each answer to form a frequency list $answer_list = \{(a_1, \text{num}_1), (a_2, \text{num}_2), \ldots, (a_m, \text{num}_m)\}$. Let a_j denote the j-th candidate answer, and n_j represent the frequency with which answer a_j appears among the k candidate chains. m is the total number of distinct answers. Based on the frequency counts, we sort answers in descending order and retain the top two with the highest vote counts, as we observe that correct answers overwhelmingly concentrate in the top two candidate answers when sampling 10 reasoning chains:

$$new_answer_list = \{(a^{(1)}, N_1), (a^{(2)}, N_2)\}$$

Let N_1 and N_2 denote the frequencies of the top two candidate answers $a^{(1)}$ and $a^{(2)}$ respectively. In exceptional cases in which several candidate answers tie for the second-highest frequency, we include all such answers in the candidate list.

For all reasoning chains associated with the top candidate answer $a^{(1)}$ and $a^{(2)}$, both forward and backward verifications are performed. Each reasoning chain is evaluated for the presence of the following three types of hallucinations: (1) condition hallucination – including condition index

errors, incorrect condition references, omissions, additions, and supplementary commonsense knowledge that is inconsistent with real-world facts. This category is assigned a weight ω_1, initially set to 0.4 in our experiments[2]. (2) Logical reasoning hallucinations – for mathematical reasoning tasks, this includes arithmetic logic errors; for linguistic reasoning tasks, it includes semantic inconsistencies. This category is assigned a weight ω_2, also set to 0.4. (3) question-comprehension hallucination – referring to cases where the reasoning chain misinterprets or fails to retain key information from the original question. This category is assigned a weight ω_3, set to 0.2.

The normalized discard score of each candidate answer is computed as follows:

$$Score_a = \frac{1}{N_a} \sum_{c \in C_a} \sum_{i=1}^{3} \omega_i \cdot \varphi_i(C)$$

$N_a = |C_a|$ denotes the number of reasoning chains associated with candidate answer a. $\varphi_t(C)$ is the indicator function for the t-th type of hallucination in the i-th reasoning chain: $\varphi_t(C) = 1$ if any intermediate reasoning step in the chain exhibits a type-t hallucination, and $\varphi_t(C) = 0$ otherwise. Finally, the candidate answer with the lower weighted discard score is selected as the final verified answer:

$$\hat{a} = \arg \min_{i \in \{1,2\}} Score_{a^{(i)}}$$

4 Experiments

In this section, we evaluate the effectiveness of CPIR method and MHCV framework across diverse tasks. We first assess individual and overall verification performance on five datasets. Then we compare final answer accuracy against strong baselines. Finally, ablation studies are conducted to demonstrate the effectiveness of each module. For experiments on voting weights and k value selection, see Appendix A.7 and A.11.

4.1 Dataset

To evaluate both arithmetic and semantic reasoning capabilities, we test our approach on several benchmark datasets. For arithmetic reasoning, we use AddSub (Hosseini et al., 2014), a dataset of simple addition and subtraction problems, GSM8K (Cobbe et al., 2021), which contains 8,500 high-quality middle school math word problems, and AQuA-RAT (Ling et al., 2017), a more challenging multiple-choice dataset covering algebra, proportions, and sequences. For semantic reasoning, we include Last Letter Concatenation, which requires extracting and concatenating the last letter of each word in a sentence to assess symbolic manipulation, and Date Understanding (DU) (Srivastava et al., 2022), which evaluates the model's ability to comprehend and reason about temporal information in date-related queries.

4.2 Experimental Preparation

Verification test of MHCV framework To evaluate the effectiveness of forward and backward verification, we manually annotated 50 reasoning chains per dataset, totaling 250 question–answer pairs. Each sample was evaluated across four verification dimensions. For each dataset, the annotated set contains 25 correct and 25 hallucinated answers, covering condition hallucinations, logical errors, question-comprehension hallucinations, and mixed-type hallucinations. A separate reliability audit of the instruction representation on 250 problems with five annotators (1,250 judgments) is provided in Appendix A.1.

Model selection GPT-3.5-Turbo serves as the primary backbone in our experiments, where both CPIR and MHCV are implemented through CoT prompting without task-specific annotations or fine-tuning. To assess the generalizability of our methods, we additionally employ DeepSeek V3-0324 as an alternative backbone (Detailed results are presented in Appendix A.11).

Model hyperparameters We set the maximum output length of LLMs to `max_tokens_len` $= 2048$ and the temperature to 0.7. During the reasoning phase, the number of candidate reasoning chains is fixed at $k = 10$. For the selection of k, we balance the inference accuracy and computational cost. The impact of k on reasoning accuracy can be found in Appendix A.11. In the Discard-Weighted Voting stage, three types of hallucination weights are assigned: $\omega_1 = 0.4$ for condition hallucination, $\omega_2 = 0.4$ for logical reasoning inconsistency, and $\omega_3 = 0.2$ for question-comprehension hallucination.

[2]The verification experiments for the weight allocation of $\omega_1, \omega_2, \omega_3$ are detailed in Appendix A.7, with theoretical discussion in Appendix A.15.

Table 1: Comparison of verification accuracy between MHCV and CoT method across five datasets for condition, logic, question-comprehension and overall hallucinations. Verified Type denotes the type of verification: condition, logic, and backward refer to accuracy in detecting condition hallucination, logical errors, and question-comprehension hallucination respectively; overall indicates the accuracy of verifying the full reasoning chain.

Verification Method	Verified Type	GSM8K	AddSub	AQuA-RAT	Last Letter	Date	Average
CoT (two-shot)	Overall	50%	58%	52%	50%	50%	52.0%
MHCV (one-shot)	Condition	64%	74%	72%	82%	72%	72.8%
	Logic	84%	90%	68%	76%	62%	76.0%
	Backward	60%	64%	76%	80%	66%	65.6%
	Overall	**68%**	**66%**	**74%**	**66%**	**60%**	**66.8%**

Table 2: Final Answer Accuracy comparison among CPIR w/o MHCV verification and three strong baselines. Each method uses GPT-3.5-Turbo as the backbone and generates 10 candidate reasoning chains per question.

Methods	Arithmetic Tasks			Semantic Tasks	
	GSM8K	AddSub	AQuA-RAT	Last Letter	Date
Self-Consistency (Wang et al., 2022)	75.74%	82.28%	59.40%	80.44%	57.99%
Faithful CoT (Lyu et al., 2023)	75.80%	88.35%	61.80%	–[3]	**73.50%**
Self-Harmonized CoT (Jin and Lu, 2024)	78.50%	87.10%	55.50%	81.30%	–[3]
Deductive Verification (Ling et al., 2024)	86.01%	93.54%	69.49%	92.60%	71.45%
CPIR (w/o Verification)	**87.57%**	**93.92%**	69.29%	**94.67%**	72.48%
CPIR + MHCV	86.81%	93.16%	**71.26%**	94.00%	71.93%

4.3 Verification Ability of MHCV Method

In this experiment, we manually annotate 50 reasoning chains per dataset following the CPIR format to assess the effectiveness of MHCV framework for each hallucination type, with 25 fully correct and 25 containing one or more errors: condition hallucination, arithmetic/semantic logic hallucination, or question-comprehension hallucination. We then evaluate whether the MHCV method can accurately identify the error type in each annotated case. We compare its overall verification performance with a standard two-shot CoT baseline to assess whether the proposed multi-dimensional framework improves verification ability.

In the evaluation of MHCV method, verification is performed across three dimensions of hallucinations. A reasoning chain is considered correct only if it successfully passes all three verification. If any single verification fails, the entire chain is deemed incorrect. In this section, we compare the MHCV framework with a two-shot CoT baseline, in which the model is guided through step-by-step verification using dedicated prompts for condition, commonsense and logical reasoning verification.

As shown in Table 1, MHCV improves overall verification accuracy over the two-shot CoT baseline by jointly targeting condition hallucinations, logical errors, and question-comprehension hallucinations. The backward verifier provides a complementary semantic signal for checking whether the reasoning trajectory preserves the original question intent, especially in longer or more complex scenarios. Since the reconstructed question is used only as a verification signal rather than a replacement for the original question, this design is supported by the instruction-reliability audit in Appendix A.1, which reports an 83.20% majority-vote pass rate and 86.00% mean pairwise agreement.

In contrast, the performance on the AddSub dataset is relatively modest. This is likely due to the dataset's inherent simplicity—both in conditional structure and computational complexity—where the advantages of condition decomposition, instruction backtracking, and multi-dimensional verification contribute less noticeably to overall reasoning quality.

4.4 Accuracy Test of CPIR and MHCV

In Section 4.3, we demonstrated that MHCV significantly enhances the verification reliability of CoT reasoning. In this section, we further evaluate the final answer accuracy of reasoning chains generated using the CPIR method and subsequently verified

[3]Faithful CoT (Lyu et al., 2023) and Self-Harmonized CoT (Jin and Lu, 2024) did not report results for the *Last Letter* and *Date* tasks, respectively; prompt formats were not specified, so the entries are left blank.

Table 3: Final answer accuracy on five datasets in the ablation study. *w/o Commonsense Enhancement* removes the commonsense-augmented module. *w/o Backtracking Verification* removes the backward verification component, with discard weights for condition hallucination and logic reasoning set to 1:1. *w/o Discard-Weighted Voting* uses Majority Voting instead.

Method	GSM8K	AddSub	AQuA-RAT	Last Letter	Date
PRISM (ours)	**86.81%**	**93.16%**	**71.26%**	**94.00%**	**71.93%**
w/o Commonsense Enhancement	84.34%	90.12%	66.54%	93.33%	69.38%
w/o Backtracking Verification	86.05%	92.66%	65.35%	93.33%	70.30%
w/o Discard-Weighted Voting	85.44%	92.15%	68.11%	**94.00%**	70.30%

through the MHCV framework. To assess final answer correctness, we adopt two evaluation settings: (1) **CPIR-only**: This setting uses the CPIR method without any verification. The final answer is selected using Majority Voting, with the number of candidate reasoning chains set to $k = 10$; (2) **CPIR + MHCV**: This setting integrates the CPIR method with the MHCV verification framework, replacing Majority Voting with the proposed Discard-Weighted Voting strategy.

We compare the performance of these two configurations against several representative methods that have demonstrated strong empirical results and were widely adopted in recent years. These include: Self-Consistency (Wang et al., 2022), Faithful CoT (Lyu et al., 2023), Self-Harmonized CoT (Jin and Lu, 2024), and Deductive Verification (Ling et al., 2024), all of which emphasize generating or validating reliable and faithful reasoning paths. Results are presented in Table 2.

After applying MHCV, we observe a slight decline in final-answer accuracy. Appendix A.2 shows that this drop is small and not statistically significant under paired bootstrap testing. Most CPIR-correct but MHCV-wrong cases are lucky-correct chains with severe hallucinations or tie-breaking instability, rather than systematic verification failures.

Interestingly, on AQuA-RAT, MHCV improves final-answer accuracy. This can be attributed to the dataset's higher rate of question-comprehension hallucination, where reverse verification better aligns reasoning with the intended question. Moreover, the dataset's complexity induces longer reasoning chains with more intermediate steps, increasing pruning during verification. In such cases, Majority Voting tends to discard more valid chains, reducing answer reliability—a limitation effectively addressed by our proposed voting strategy.

4.5 Ablation Experiment

We conduct ablation studies to evaluate the contribution of each MHCV component. As shown in Table 3, we examine three variants: (1) w/o Commonsense Enhancement; (2) w/o Backward Verification; and (3) w/o Discard-Weighted Voting. In the last setting, if all candidate chains are rejected or multiple answers tie, the model outputs the answer with the highest vote count.

We observe that the AQuA-RAT dataset exhibits the most substantial performance drop, likely due to its complex reasoning structures and higher incidence of question-comprehension hallucination. Moreover, applying Majority Voting within multi-dimensional verification often leads to a limited pool of valid voting samples, thereby reducing the reliability of the final answer selection.

5 Conclusions

This study aims to improve question comprehension, intermediate reasoning quality, and answer selection reliability in Chain-of-Thought through a multi-dimensional perspective. To achieve this, we propose a two-stage framework, **PRISM**. The first stage leverages LLMs' own capabilities to enhance commonsense and generate progressive instructions to support subsequent verification. The second stage introduces the Multi-Dimensional Heterogeneous Collaborative Verification (MHCV) framework to detect condition, logic, and question-comprehension hallucinations. Additionally, we design the Discard-Weighted Voting strategy to support answer selection under multi-dimensional filtering. Experiments show that PRISM improves question-comprehension ability and reasoning-chain reliability while maintaining competitive final-answer accuracy. Our approach provides a multi-stage and multi-dimensional framework for future chain-of-thought verification studies.

Limitations

While our method effectively mitigates multi-dimensional hallucinations, three limitations remain:

(1) As shown in Table 1, verification of question-comprehension hallucinations lags behind other checks. This gap is attributable to the volatility of semantic scores generated by LLMs in backward verification. Future work could further explore the setting of varying τ and conduct more experiments using a broader range of LLMs. For a detailed analysis, see appendix A.14.

(2) We currently apply a unified framework across various tasks, but task-specific adaptations may yield better performance. For example, AQuA-RAT is more prone to question-comprehension hallucination, while others rely more on commonsense or instruction-level cues. Thus, optimal strategies and voting weights may vary by task and warrant further exploration.

(3) Our current experiments are conducted on relatively standard reasoning benchmarks, and do not yet cover more challenging settings such as MATH datasets with substantially deeper and more compositional reasoning. In addition, the manually annotated hallucination set used for analysis is still limited in size. In future work, we plan to evaluate PRISM on more complex benchmarks, expand the manually annotated dataset, and perform a more fine-grained statistical analysis of different hallucination categories so that the annotation can better match realistic task distributions.

Ethical Considerations

This work focuses on improving the reliability of chain-of-thought reasoning by mitigating different types of hallucinations in large language models. The proposed framework does not introduce new data sources, user profiling, or deployment-specific assumptions, and all experiments are conducted on publicly available benchmark datasets. As such, no direct risks related to privacy, data misuse, or human subjects are involved.

By enhancing question comprehension and intermediate reasoning verification, this work aims to reduce misleading or incorrect model outputs, which may help improve the safety and trustworthiness of language models in downstream applications. Nevertheless, the framework relies on the reasoning capabilities of existing large language models and inherits their limitations and potential biases. Care should be taken when applying this method in high-stakes or real-world settings, where incorrect reasoning could have significant consequences.

References

Itai Arcuschin, Jakub Janiak, Robert Krzyzanowski, and 1 others. 2025. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*.

Y. Chen, H. Deng, K. Han, and 1 others. 2025. Policy frameworks for transparent chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2503.14521*.

Xinyu Cheng, Jiawei Li, Wayne Xin Zhao, and 1 others. 2024. Chainlm: Empowering large language models with improved chain-of-thought prompting. *arXiv preprint arXiv:2403.14312*.

Zheng Chu, Jinduo Chen, Qian Chen, and 1 others. 2024. Navigate through enigmatic labyrinth: A survey of chain of thought reasoning: Advances, frontiers and future. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1173–1203.

Karl Cobbe, Vineet Kosaraju, Megha Bavarian, Jacob Hilton, Rewon Child, Mark Chen, Heewoo Jun, Christopher Hesse, John Schulman, Jacob Steinhardt, and Greg Brockman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Aniruddha Deb, Neeva Oza, Sarthak Singla, Dinesh Khandelwal, Dinesh Garg, and Parag Singla. 2024. Fill in the blank: Exploring and enhancing llm capabilities for backward reasoning in math word problems. *arXiv preprint arXiv:2310.01991*.

G. Feng, B. Zhang, Y. Gu, and 1 others. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Kai Hu, Ziyi Chen, Chung-Hsien Hsia Yang, and 1 others. 2025. Chain-of-thought prompting for speech translation. In *ICASSP 2025 - IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

L. Huang, W. Yu, W. Ma, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

T. Inaba, H. Kiyomaru, F. Cheng, and 1 others. 2023. Multitool-cot: Gpt-3 can use multiple external tools with chain of thought prompting. *arXiv preprint arXiv:2305.16896*.

F. Jiang, Z. Xu, Y. Li, and 1 others. 2025. Safechain: Safety of language models with long chain-of-thought reasoning capabilities. *arXiv preprint arXiv:2502.12025*.

W. Jiang, H. Shi, L. Yu, and 1 others. 2024. Forward-backward reasoning in large language models for mathematical verification. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6647–6661.

Zhiyang Jin and Wei Lu. 2024. Self-harmonized chain of thought. *arXiv preprint arXiv:2409.04057*.

J. Jung, L. Qin, S. Welleck, and 1 others. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. *arXiv preprint arXiv:2205.11822*.

A. Kumar, H. Kim, J. S. Nathani, and 1 others. 2025. Improving the reliability of llms: Combining cot, rag, self-consistency, and self-verification. *arXiv preprint arXiv:2505.09031*.

Jiawei Li, Guangyu Li, Yanyan Li, and 1 others. 2025. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23.

Hao Liao, Hong Kong, Bowen Wang, and 1 others. 2025. Cot-drive: Efficient motion forecasting for autonomous driving with llms and chain-of-thought prompting. *IEEE Transactions on Artificial Intelligence*.

Wang Ling, Dani Yogatama, Chris Dyer, William W. Cohen, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Z. Ling, Y. Fang, X. Li, and 1 others. 2024. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36.

Q. Lyu, S. Havaldar, A. Stein, and 1 others. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Ankur Madaan, Niket Tandon, Prakhar Gupta, and 1 others. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

N. Miao, Y. W. Teh, and T. Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *arXiv preprint arXiv:2308.00436*.

Chirag Mitra, Bowen Huang, Trevor Darrell, and 1 others. 2024. Compositional chain-of-thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14431.

X. Ni, Q. Wang, Y. Zhang, and 1 others. 2025. Toolfactory: Automating tool generation by leveraging llm to understand rest api documentations. *arXiv preprint arXiv:2501.16945*.

Yuxuan Nong, Mohammed Aldeen, Lianliang Cheng, and 1 others. 2024. Chain-of-thought prompting of large language models for discovering and fixing software vulnerabilities. *arXiv preprint arXiv:2402.17230*.

D. Paul, M. Ismayilzada, M. Peyrard, and 1 others. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

J. Qian, H. Wang, Z. Li, and 1 others. 2022. Limitations of language models in arithmetic and symbolic induction. *arXiv preprint arXiv:2208.05051*.

Leonardo Ranaldi and Giulia Pucci. 2025. Exploring backward reasoning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*.

M. Sanwal. 2025. Layered chain-of-thought prompting for multi-agent llm systems: A comprehensive approach to explainable large language models. *arXiv preprint arXiv:2501.18645*.

Aarohi Srivastava, Jason Phang, Xavier Garcia, and et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

R. Vacareanu, A. Pratik, E. Spiliopoulou, and 1 others. 2024. General purpose verification for chain of thought prompting. *arXiv preprint arXiv:2405.00204*.

K. Wang, J. P. Zhou, J. Chang, and 1 others. 2025. Value-guided search for efficient chain-of-thought reasoning. *arXiv preprint arXiv:2505.17373*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

M. Wu, T. Zhu, H. Han, and 1 others. 2025. Chain-of-tools: Utilizing massive unseen tools in the cot reasoning of frozen language models. *arXiv preprint arXiv:2503.16779*.

Y. Wu, Z. Zhang, and H. Zhao. 2024. Mitigating misleading chain-of-thought reasoning with selective filtering. *arXiv preprint arXiv:2403.19167*.

E. Zelikman, Y. Wu, and N. D. Goodman. 2022. Star: Self-taught reasoner. In *Proceedings of the NIPS*, volume 22.

D. Zhang, J. Yang, H. Lyu, and 1 others. 2024. Cocot: Contrastive chain-of-thought prompting for large multimodal models with multiple image inputs. *arXiv preprint arXiv:2401.02582*.

S. Zhang, Y. Dong, J. Zhang, and 1 others. 2025. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

S. Zhang and D. Xiong. 2025. Backmath: Towards backward reasoning for solving math problems step by step. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 466–482.

Y. Zhang, H. Cai, X. Song, and 1 others. 2023. Reverse chain: A generic-rule for llms to master multi-api planning. *arXiv preprint arXiv:2310.04474*.

R. Zhao, X. Li, S. Joty, and 1 others. 2023. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. *arXiv preprint arXiv:2305.03268*.

D. Zhou, N. Schärli, L. Hou, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

T. Zhu, K. Zhang, J. Xie, and 1 others. 2024. Deductive beam search: Decoding deducible rationale for chain-of-thought reasoning. *arXiv preprint arXiv:2401.17686*.

A Appendix

A.1 Reliability of Progressive Instruction Lists

The reverse instructional backtracking module in MHCV relies on the progressive instruction representation extracted from CPIR-style reasoning chains. A potential concern is that this representation may be too lossy to support faithful question reconstruction, especially after removing detailed intermediate reasoning and numerical surface forms. To examine this issue directly, we conduct a stand-alone reliability study on 250 question–answer records, with 50 samples from each benchmark dataset used in the main experiments.

For each CPIR reasoning chain, we remove the detailed intermediate reasoning content, the final answer, and all answer markers. We retain only the compact number-masked condition–instruction representation, including the label-rewritten condition statements and the progressive instruction titles. All concrete numerical values in the condition statements are replaced by `<NUM>`. Therefore, the reconstruction model cannot rely on exact numerical values or final answers; instead, it must infer the original question intent from the abstracted entities, relations, operation sequence, and target quantity encoded by the extracted representation.

We then reconstruct a self-contained natural-language question from this compact representation. Each reconstructed question is independently evaluated by five human annotators using the same annotation guideline. Each of the 250 problems is evaluated by all five annotators, resulting in 1,250 judgment records. Each annotator provides five scoring fields: (1) `semantic_score`, a 1–5 semantic similarity score; (2) `matches_original`, indicating whether the reconstructed question matches the original question intent; (3) `solvable_same_answer`, indicating whether the reconstructed question would lead to the same answer; (4) `instruction_clarity`, indicating whether the extracted instruction representation is clear; and (5) `instruction_completeness`, indicating whether the representation contains sufficient information for reconstruction.

We use majority voting over `matches_original` as the primary pass criterion. In addition, we report same-answer preservation, semantic similarity, instruction clarity, instruction completeness, and inter-annotator agreement. Same-answer preservation is particularly important for our setting because reverse verification should not only recover a semantically plausible question, but also preserve the answer-relevant intent of the original problem.

To quantify uncertainty, we report bootstrap confidence intervals for the primary majority-vote pass rate. Specifically, we use 100,000 stratified non-parametric bootstrap resamples. For dataset-level estimates, records are sampled with replacement within each dataset. For the overall estimate, we preserve the balanced 50-sample-per-dataset design by resampling within each dataset and then aggregating the resampled records across all five datasets. This avoids over-interpreting dataset-level percentages under the fixed 50-sample setting.

As shown in Table 4, the extracted number-masked condition–instruction representation achieves an overall majority-vote pass rate of 83.20%, corresponding to 208 successful reconstructions out of 250 problems. The stratified bootstrap 95% confidence interval is [78.8%, 87.6%], indicating that the observed reliability is not driven by a small number of isolated cases. The same-answer majority rate is also 83.20%, showing that the reconstructed questions judged as aligned with the original intent are generally also judged to preserve answer recoverability. The mean semantic score over all 1,250 judgments is 4.534 out of 5, suggesting that most reconstructions are semantically equivalent to, or highly aligned with, the original questions.

At the dataset level, Last Letter obtains a 100.00% majority pass rate because its instruction trajectory directly specifies the symbolic operation required by the task. GSM8K, Date Understanding, and AddSub also show high majority pass rates, reaching 86.00%, 84.00%, and 82.00%, respectively. AQuA-RAT obtains a lower pass rate of 64.00%, and its Same-Ans. rate is also 64.00%. This is consistent with the algebraic and multiple-choice structure of AQuA-RAT: when numerical values and answer options are abstracted away, reconstructing the exact option semantics and quantity relations becomes more difficult. Therefore, the dataset-level results should be interpreted as diagnostic evidence, while the overall 250-problem majority-vote result is the primary reliability indicator.

Table 5 reports the per-annotator statistics for the five human annotators. The mean per-annotator match rate is 80.32%, and the mean same-answer rate is 80.48%. The close alignment between

Table 4: Reliability of the extracted number-masked condition–instruction representation for reverse question reconstruction. Majority Pass is computed over five human annotators for each problem. CI denotes the stratified bootstrap 95% confidence interval for the majority-vote pass rate. Same-Ans. denotes the majority-vote rate for `solvable_same_answer`.

Dataset	#Prob.	#Judg.	Maj. Pass	Maj. Rate	95% CI	Same-Ans.	Sem.
AddSub	50	250	41 / 50	82.00%	[70.0%, 92.0%]	82.00%	4.432
AQuA-RAT	50	250	32 / 50	64.00%	[50.0%, 76.0%]	64.00%	4.188
Date Understanding	50	250	42 / 50	84.00%	[74.0%, 94.0%]	84.00%	4.516
GSM8K	50	250	43 / 50	86.00%	[76.0%, 94.0%]	86.00%	4.532
Last Letter	50	250	50 / 50	100.00%	[100.0%, 100.0%]	100.00%	5.000
Overall	**250**	**1,250**	**208 / 250**	**83.20%**	**[78.8%, 87.6%]**	**83.20%**	**4.534**

Table 5: Per-human-annotator reliability statistics over the same 250 reconstruction problems. Each annotator evaluates all problems using the same five-field schema.

Annotator	Match Rate	Same-Ans. Rate	Mean Sem.	Median Sem.	Clarity-Yes	Completeness-Yes
H1	74.40%	74.80%	4.524	5.0	84.40%	64.80%
H2	87.60%	87.60%	4.744	5.0	100.00%	86.40%
H3	74.80%	74.80%	4.224	4.0	99.60%	62.00%
H4	85.60%	85.60%	4.680	5.0	99.60%	82.40%
H5	79.20%	79.60%	4.496	5.0	99.20%	72.80%
Mean	**80.32%**	**80.48%**	**4.534**	–	**96.56%**	**73.68%**

these two rates suggests that semantic intent matching and answer preservation are strongly correlated in this evaluation. At the same time, the annotator-level results show natural variation in human judgments: match rates range from 74.40% to 87.60%, and same-answer rates range from 74.80% to 87.60%. This variation indicates that the reliability conclusion is not driven by a single annotator, but is supported across the full annotation panel.

Table 6 further analyzes the five judgment fields. For `matches_original`, 80.32% of all individual judgments are positive. For `solvable_same_answer`, the positive rate is 80.48%, which is almost identical to the semantic match rate. This result is important for reverse verification: it suggests that when the compact instruction representation preserves the original question intent, it usually also preserves the answer-relevant structure of the task. Conversely, the remaining negative Same-Ans. judgments reveal cases where semantic reconstruction alone is insufficient, especially when exact quantities, option semantics, or temporal offsets are necessary for recovering the same answer.

For `semantic_score`, 66.32% of judgments receive the maximum score of 5, and another 21.04% receive a score of 4, meaning that 87.36% of all judgments are high-similarity cases. This supports the use of the reconstructed question as a semantic signal in MHCV. The clarity and completeness fields provide a finer diagnosis of the extracted representation. The instruction representation is judged clear in 96.56% of all records, suggesting that the extracted instruction lists are generally understandable to human annotators. Completeness is more conservative: 73.68% of records are judged fully complete, while 26.16% are partially complete. This distinction indicates that most reconstruction failures are not caused by unclear instructions, but by deliberate information abstraction, especially numerical masking and the removal of fine-grained option or temporal details.

As shown in Table 7, the five human annotators achieve a mean pairwise agreement of 86.00% on `matches_original`, with pairwise agreement ranging from 82.80% to 90.00%. Fleiss' κ is 0.5572 for `matches_original` and 0.5621 for `solvable_same_answer`. Since positive labels are frequent in this evaluation, chance-corrected agreement is more conservative than raw pairwise agree-

Table 6: Distribution of the five scoring fields across all 1,250 human judgment records.

Scoring Field	Positive / High	Partial / Medium	Negative / Low
matches_original	Yes: 1,004 / 1,250 (80.32%)	–	No: 246 / 1,250 (19.68%)
solvable_same_answer	Yes: 1,006 / 1,250 (80.48%)	–	No: 244 / 1,250 (19.52%)
semantic_score	Score 5: 829 / 1,250 (66.32%)	Score 4: 263 / 1,250 (21.04%)	Score ≤ 3: 158 / 1,250 (12.64%)
instruction_clarity	Yes: 1,207 / 1,250 (96.56%)	Partial: 43 / 1,250 (3.44%)	No: 0 / 1,250 (0.00%)
instruction_completeness	Yes: 921 / 1,250 (73.68%)	Partial: 327 / 1,250 (26.16%)	No: 2 / 1,250 (0.16%)

Table 7: Inter-annotator agreement for human judgments.

Metric	Value
Number of problems	250
Number of annotators	5
Mean pairwise agreement on matches_original	86.00%
Pairwise agreement range	82.80%–90.00%
Fleiss' κ on matches_original	0.5572
Fleiss' κ on solvable_same_answer	0.5621

ment. Overall, the agreement analysis indicates that the reliability result is stable across human annotators.

We also analyze whether longer instruction lists become less reliable. As shown in Table 8, the reconstruction pass rate remains generally stable across different instruction lengths. For the most frequent buckets, namely 1–4 instructions, the majority-vote pass rate ranges from 76.09% to 86.76%. There is no monotonic degradation as instruction length increases, suggesting that the progressive instruction representation remains robust even when the reasoning trajectory contains multiple steps.

Finally, we inspect the 42 majority-failure cases. The failures are distributed as follows: 18 in AQuA-RAT, 9 in AddSub, 8 in Date Understanding, 7 in GSM8K, and 0 in Last Letter. Most failures arise from one of three sources: (1) some quantity relations become under-specified after numerical masking; (2) multiple-choice option semantics in AQuA-RAT are difficult to reconstruct exactly from abstracted instructions; and (3) temporal offsets in Date Understanding can be shifted when concrete date values are removed. These cases explain why Same-Ans. is a necessary complement to semantic matching: a reconstruction may be close in wording or intent while still failing to preserve the precise answer-relevant structure. Therefore, the extracted number-masked condition–instruction representation provides a reliable semantic and procedural signal for reverse instructional backtracking, while MHCV still treats it as a verification signal rather than as a replacement for the original question.

A.2 Accuracy–Reliability Trade-off after Multi-Dimensional Verification

In the main results, applying MHCV leads to a slight decrease in final-answer accuracy on several datasets. This raises a natural question: does multi-dimensional verification systematically harm answer accuracy, or does the decrease reflect a trade-off between answer-level voting and reasoning-chain reliability? To answer this question, we conduct an additional paired analysis comparing CPIR-only majority voting and CPIR+MHCV on the same test instances.

For each problem, we record whether the CPIR-only answer and the CPIR+MHCV answer are correct. This gives four transition types: **CC**, where both methods are correct; **CW**, where CPIR-only is correct but MHCV is wrong; **WC**, where CPIR-only is wrong but MHCV is correct; and **WW**, where both methods are wrong. We further conduct a manual regression audit over the CW cases and evaluate a conservative MHCV variant that only overrides the CPIR-only majority-vote answer when the MHCV-preferred candidate has a sufficiently lower discard score. The margin threshold is denoted by γ.

As shown in Table 9, the strict MHCV setting changes only a small fraction of answer outcomes. Among 2,485 paired instances, CPIR+MHCV corrects 37 CPIR-only errors, while 56 originally correct CPIR-only answers become incorrect after verification. The resulting accuracy change is -0.76%. Under paired bootstrap testing, the 95% confidence interval is $[-1.53\%, 0.00\%]$, and the decrease is not statistically significant. This suggests that the observed answer-level drop is small and should not be interpreted as a systematic failure of multi-dimensional verification.

Table 10 further analyzes why these 56 regres-

Table 8: Effect of instruction-list length on reconstruction reliability. Pass rates are computed by majority vote over five human annotators. Buckets with fewer than five examples are omitted to avoid over-interpreting extremely small groups.

#Instructions	#Problems	Majority Pass	Mean Semantic Score
1	33	81.82%	4.485
2	136	86.76%	4.610
3	46	76.09%	4.404
4	23	78.26%	4.504
5–6	9	88.89%	4.422

Table 9: Paired transition analysis from CPIR-only to strict CPIR+MHCV. CC denotes both methods correct; CW denotes CPIR-only correct but MHCV wrong; WC denotes CPIR-only wrong but MHCV correct; WW denotes both methods wrong. ΔAcc is computed as $\mathrm{Acc}_{\mathrm{MHCV}} - \mathrm{Acc}_{\mathrm{CPIR}}$. CI is obtained by paired bootstrap over test instances.

Comparison	#Inst.	CC	CW	WC	WW	ΔAcc.	95% CI
CPIR-only $\rightarrow$ CPIR+MHCV	2,485	2,054	56	37	338	-0.76%	$[-1.53\%, 0.00\%]$

Table 10: Manual audit of the 56 CW regression cases, where CPIR-only is correct but strict MHCV is wrong.

Regression Cause	#Cases	Ratio
Severe hallucination with lucky answer	31	55.36%
Tie-breaking instability	16	28.57%
Harmless reasoning flaw	1	1.79%
Backward semantic false positive	2	3.57%
Unclassified	6	10.71%
Total	**56**	**100.00%**

sions occur. More than half of them, 31 out of 56 cases, are severe-hallucination lucky-answer cases: the CPIR-only final answer is correct, but the supporting reasoning chain contains a severe condition, logic, or question-comprehension hallucination. These cases indicate that final-answer accuracy alone may overestimate reasoning reliability, because an answer can be correct even when the reasoning path is not faithful. Another 16 cases are caused by tie-breaking instability among close candidate answers, suggesting that part of the regression comes from the answer-selection policy rather than from the verification signal itself. Only a small number of cases are due to harmless reasoning flaws or backward semantic false positives.

To further test whether the slight accuracy drop is caused by an overly aggressive selection policy, we evaluate a conservative MHCV variant. Instead of always selecting the answer with the lowest discard score, conservative MHCV starts from the CPIR-only majority-vote answer and overrides it only when the MHCV-preferred answer has a sufficiently large discard-score advantage. Formally, let a_{MV} be the CPIR-only majority-vote answer and a_{MHCV} be the answer preferred by MHCV. The conservative rule is:

$$\hat{a} = \begin{cases} a_{\mathrm{MHCV}}, & \text{if } Score(a_{\mathrm{MV}}) - Score(a_{\mathrm{MHCV}}) > \gamma, \\ a_{\mathrm{MV}}, & \text{otherwise.} \end{cases}$$

Here, γ controls the strength of the evidence required to override majority voting. A larger γ makes the method more conservative.

Table 11 shows that the answer-level accuracy drop is controllable. The strict MHCV setting achieves the lowest average discard score, reducing Avg. Discard from 0.2177 to 0.1373, but it also overrides 41.99% of CPIR-only majority-vote answers. This aggressive intervention explains the small decrease in final-answer accuracy. In contrast, the conservative variant with $\gamma = 0.00$ reaches 84.99% final-answer accuracy, slightly higher than CPIR-only at 84.91%, while still reducing Avg. Discard from 0.2177 to 0.1816. This corresponds to a 16.58% relative reduction in discard score:

$$\frac{0.2177 - 0.1816}{0.2177} = 16.58\%.$$

Therefore, a simple margin-based fallback can preserve answer accuracy while retaining part of the reliability benefit introduced by MHCV.

As γ increases, the override rate decreases from 15.15% to 3.90%, and the method gradually approaches CPIR-only majority voting. This trend

Table 11: Conservative MHCV sweep under different discard-score margins γ. CW denotes CPIR-correct but method-wrong cases, and WC denotes CPIR-wrong but method-correct cases. Override denotes the percentage of instances in which the CPIR-only majority-vote answer is replaced.

Method	**Final Acc.**	**CW**	**WC**	**Override**	**Avg. Discard**
CPIR-only	84.91%	0	0	0.00%	0.2177
MHCV (strict)	84.14%	56	37	41.99%	0.1373
MHCV, $\gamma = 0.00$	84.99%	26	28	15.15%	0.1816
MHCV, $\gamma = 0.05$	84.83%	25	23	12.12%	0.1823
MHCV, $\gamma = 0.10$	84.83%	23	21	10.39%	0.1837
MHCV, $\gamma = 0.15$	84.71%	23	18	9.09%	0.1851
MHCV, $\gamma = 0.20$	84.51%	22	12	5.63%	0.1913
MHCV, $\gamma = 0.25$	84.47%	22	11	5.19%	0.1922
MHCV, $\gamma = 0.30$	84.43%	21	9	3.90%	0.1957

confirms that MHCV provides a tunable accuracy–reliability trade-off. The strict setting is reliability-oriented and aggressively minimizes hallucination-related discard scores, whereas the conservative setting is accuracy-preserving and only uses MHCV when the verification evidence is strong enough. Overall, these results show that the slight accuracy drop reported in the main experiments is not caused by a systematic weakness of multi-dimensional verification. Rather, it reflects the choice of an aggressive verification-based answer-selection policy. When a conservative fallback is used, MHCV can recover final-answer accuracy while still selecting reasoning supports with lower verification risk.

A.3 Verification-Time Token Cost Comparison

To quantify the computational cost of different chain-of-thought verification strategies, we compare four methods on the same 250 manually annotated reasoning-chain records. For each method, we report the number of evaluated samples, the number of LLM API calls, the total token usage recorded in the experimental logs, the average token cost per sample, and the average token cost per API call. This experiment focuses on the verification-time cost under a unified evaluation set, rather than on full benchmark-level inference cost. This comparison isolates post-hoc verification cost; the end-to-end cost of PRISM also includes generating k CPIR candidate chains. Therefore, the results should be interpreted as a verification-time efficiency comparison under comparable verification coverage, rather than evidence that PRISM is the lowest-cost full pipeline.

As shown in Table 12, Deductive Verification has the highest total token cost, consuming 1,446,819 tokens in total and 5,787.28 tokens per sample on average. Although it mainly follows a forward verification paradigm, its verification process requires repeated judgments over decomposed conditions, intermediate conclusions, and the current reasoning step to be verified. For each candidate reasoning chain, the model often needs to repeatedly take the original question, the condition set, previously derived intermediate conclusions, and the current step as input to determine whether the step can be entailed by the available premises. Therefore, as the length of the reasoning chain increases, the verification context is repeatedly carried across steps, resulting in a relatively high total token cost.

Our proposed Multi-Dimensional Heterogeneous Collaborative Verification method, denoted as MHCV, consumes 1,411,792 tokens in total and 5,647.17 tokens per sample on average. Its total token cost is slightly lower than that of Deductive Verification, reducing the cost by 35,027 tokens, corresponding to a relative reduction of 2.42%.

This result is notable because MHCV performs both forward and backward verification and covers three types of hallucinations, namely condition hallucination, logical hallucination, and question-comprehension hallucination. Although MHCV invokes the LLM more frequently, its average token cost per API call is only 644.07 tokens, substantially lower than that of Deductive Verification. For MHCV, each API call processes all candidate reasoning chains simultaneously for a given verification dimension, so the 2,192 calls reflect multiple verification passes and pre-processing steps across 250 samples rather than one call per chain per dimension. This is because the backward verification

Table 12: Overall token cost comparison of four chain-of-thought verification strategies on the same 250 annotated reasoning-chain records. Total Tokens denotes the API-reported total token usage over all calls. Avg. Tokens / Sample is computed by dividing total tokens by 250, and Avg. Tokens / Call is computed by dividing total tokens by the number of API calls.

Method	#Samples	#API Calls	Total Tokens	Avg. Tokens / Sample	Avg. Tokens / Call
Deductive Verification	250	1,365	1,446,819	5,787.28	1,059.94
Faithful CoT	250	300	599,384	2,397.54	1,997.95
Self-Harmonized CoT	250	750	1,163,834	4,655.34	1,551.78
MHCV (ours)	250	2,192	1,411,792	5,647.17	644.07

stage does not reconstruct the complete problem conditions or a new full reasoning path. Instead, it uses the progressive instruction representation to perform question-semantic backtracking, which reduces the long-context overhead typically introduced by backward verification. Meanwhile, the forward verification stage is organized around condition consistency and logical consistency, giving each verification call a clearer and more bounded input scope.

Faithful CoT has the lowest total token cost among the four methods, consuming 599,384 tokens in total and 2,397.54 tokens per sample on average. This is expected because Faithful CoT focuses more on producing faithful and structured reasoning processes rather than performing fine-grained, multi-dimensional post-hoc verification over candidate chains. It does not separately verify condition hallucination, logical hallucination, and question-comprehension hallucination, nor does it require a reverse question-backtracking procedure. Consequently, both its number of API calls and total token consumption are relatively low. However, its verification coverage is also more limited, making it less directly suited for identifying subtle question-understanding deviations inside reasoning chains.

Self-Harmonized CoT consumes 1,163,834 tokens in total and 4,655.34 tokens per sample on average, which is lower than MHCV and Deductive Verification but higher than Faithful CoT. This method typically improves answer stability by exploiting consistency or harmonization among multiple candidate reasoning chains. Therefore, it requires more candidate generation and comparison than Faithful CoT, leading to higher token consumption. Nevertheless, because it does not explicitly decompose verification into condition, logic, and question-understanding dimensions, its overall token cost remains lower than that of MHCV.

Overall, MHCV incurs higher token cost than Faithful CoT and Self-Harmonized CoT because it aims at broader verification coverage. In particular, it simultaneously performs forward condition and logic verification as well as backward question-comprehension verification. However, compared with Deductive Verification, which also conducts fine-grained reasoning checks, MHCV achieves a slightly lower total token cost through its progressive instruction-based backward verification design. These results show that MHCV introduces multi-dimensional hallucination verification while keeping the computational overhead within a reasonable range, achieving a practical balance between verification coverage and resource consumption.

A.4 Computational Cost of Backward Verification

Backward verification cost analysis Since the proposed MHCV framework integrates both forward and backward verification to detect condition hallucinations, logical hallucinations, and question-comprehension hallucinations, its computational cost is inherently higher than that of conventional verification schemes. In particular, backward verification requires reconstructing the original conditions, question semantics, and an additional set of reasoning instructions, resulting in a substantial increase in token usage. Therefore, this section compares the token cost of our instruction-based backward reconstruction method with two baseline backward-verification approaches.

We include two backward verification baselines:

(1) *Forward-backward verification (Jiang et al., 2024)*, which masks a single condition and infers it from the final answer to determine answer correctness;

(2) *Full backward reconstruction*, which utilizes the entire reasoning chain to infer the semantic content of the original question and generate a full-

Table 13: Token consumption and backward verification accuracy comparison across three backward reasoning methods on the same 250 annotated reasoning-chain records. Total Tokens denotes the API-reported total token usage over all calls. *Full Backward* reconstructs the entire original question using the full reasoning chain. *Forward–Backward Reasoning* masks a key condition and infers it backward from the final answer. *MHCV (ours)* employs progressive instruction-based backward reconstruction.

Method	**Prompt Tokens**	**Output Tokens**	**Total Tokens**	**Accuracy**
Full Backward	199,051	28,919	227,970	66.40%
Forward–Backward Reasoning	123,051	46,110	169,161	73.60%
MHCV (ours)	110,801	23,942	134,743	79.20%

step instructional trace.

In the experiment, we use the manually annotated set of 250 candidate reasoning chains, focusing exclusively on evaluating backward question-understanding verification. Unlike previous datasets, this set does not include condition or logical hallucination cases. Consequently, no interference occurs between different hallucination types, and the backward verification accuracy of MHCV is expected to be higher than in earlier evaluations. Table 13 report the total token consumption and backward verification accuracy of the three methods on the annotated dataset.

Results and analysis The results show that reconstructing the original question from the full reasoning trace significantly increases the number of prompt tokens, leading to a total token cost approximately 69.19% higher than our method. Moreover, providing excessive reasoning-chain information does not improve verification accuracy; instead, it reduces accuracy, suggesting that effective backward reconstruction depends on step-wise instructional cues rather than full-chain reasoning content.

For the forward–backward reasoning method, its verification target differs from ours: it infers masked conditions rather than reconstructing question semantics. Although it requires even more tokens than our approach, its accuracy is noticeably lower. These findings demonstrate that our instruction-based backward reconstruction method achieves a more favorable balance between efficiency and accuracy, outperforming both baselines in token cost and verification reliability.

A.5 Algorithmic Details of Backtracking Verification

Algorithm 1: Reverse Instructional Problem Backtracking Verification

Input: Original question q, decomposed question $Q' = \{b_1, b_2, \ldots, b_m, q\}$, commonsense knowledge C
Output: Verification result (Pass / Fail)

Let $B \leftarrow Q' \setminus \{q\}, \quad I \leftarrow \emptyset$
for $t = 1$ to k **do**
    Generate a candidate reasoning chain $R_t = \{(i, g, r)\}$ using prompt $(B \cup C)$
    Extract instruction list $I_t \leftarrow \{i \mid (i, g, r) \in R_t\}$
    $I \leftarrow I \cup I_t$
end for
Generate predicted question $q' \leftarrow$ LLM.generate$(I, B \cup C)$
score $\leftarrow$ LLM.evaluate$(q', q) \in [0, 1]$
Let τ be a validation-tuned similarity threshold, set to 0.8 in our experiment
if score $> \tau$ **then**
    return Pass
else
    return Fail
end if

A.6 Performance of PRISM on Open Source LLM

In addition to evaluating the inference and verification capabilities of the proprietary model GPT-3.5-turbo on five benchmark datasets, we further examine the generalizability of PRISM by incorporating a recent popular open-source language model, Deepseek V3-0324. During both the reasoning and verification stages, responses are generated using Deepseek V3-0324, adhering to the Progressive Instructional Reasoning format illustrated in Figure 1.

As shown in Table 14, Deepseek V3-0324 demonstrates notable improvements in accuracy

Table 14: Comparative performance of Deepseek V3-0324 and GPT-3.5-turbo on five reasoning datasets under PRISM framework.

Model Name	GSM8K	AddSub	AQuA-RAT	Last Letter	Date Understanding
Deepseek V3-0324	96.21%	96.96%	88.98%	100.00%	91.33%
GPT-3.5-turbo	86.81%	93.16%	71.26%	94.00%	71.93%

across all five datasets, attributed to its strong arithmetic and semantic reasoning capabilities. The performance gains are especially pronounced on more challenging tasks, further validating the generalizability of PRISM in both the reasoning and verification stages under the chain-of-thought paradigm.

A.7 Final Answer's Accuracy of Different Discard Weights

As shown in Table 15, the first three rows present configurations where each type of hallucination is individually assigned a dominant weight. These settings yield lower final answer accuracy on most datasets compared to the 2:2:1 ratio, suggesting that the three verification types are complementary and that overemphasis on any single type weakens overall validation. The fourth row evaluates an equal-weight setting (1:1:1), which also underperforms, indicating that the three types contribute unequally to final answer quality and highlighting the benefit of strategically weighted verification.

An exception is observed in the Last Letter dataset, where certain configurations yield higher final answer accuracy. This is likely due to its shallow reasoning structure, which involves surface-level text extraction without complex logic or implicit commonsense. As a result, condition and logic hallucination are rare, while question-comprehension hallucination—such as mistaking the task for extracting first or middle letters—are more common. In this context, assigning greater weight to backward verification helps correct such misinterpretations and improves final answer accuracy.

A.8 Dataset Details and Final Answer Extraction

To comprehensively evaluate the reasoning and verification capabilities of PRISM across a range of tasks, we conduct experiments on five benchmark datasets. Specifically, GSM8K, AddSub, and AQuA-RAT are employed to assess arithmetic reasoning, while Date Understanding and Last Letter are used to evaluate semantic logical reasoning. A detailed overview of each dataset, including sample size, the format of answers, and type of task, is provided in Table 16.

Due to the varying answer formats across datasets, we standardized the model outputs to ensure consistent and accurate final answer extraction. To this end, we instructed the model to append a summary sentence containing the final answer, following a designated delimiter ####. This approach facilitates reliable parsing of the final answer. A representative code snippet for the extraction function is shown in Table 17.

A.9 Backward Verification of question-comprehension hallucination

Question-comprehension hallucination refers to a model's failure to accurately comprehend the true intent of a question. This represents a deeper and more nuanced form of hallucination, often challenging to detect through conventional chain-of-thought (CoT) verification methods. There are two primary scenarios in which this type of hallucination tends to occur:

- **Semantic Misinterpretation:** The model initially fails to capture the correct semantic meaning of the question, resulting in a misinterpretation of what is actually being questioned.

- **Contextual Drift:** As the reasoning process unfolds, the model gradually loses focus on the original question or forgets its content altogether, causing its reasoning trajectory to deviate from the intended objective.

In comparison to condition hallucination and logic hallucination, question-comprehension hallucination are more deeply embedded and significantly harder to detect—even with a dedicated verification module. This form of error often stems

Table 15: Impact of different weight configurations in Discard-Weighted Voting on final answer accuracy across five datasets. dc:dl:db = 2:2:1 corresponds to weights of 0.4 (condition), 0.4 (logic), and 0.2 (backward). The remaining configurations are scaled proportionally. The last row (in bold) represents the configuration adopted in our main experiments.

Weight Distribution	GSM8K	AddSub	AQuA-RAT	Last Letter	Date
dc:dl:db = 8:1:1	86.43%	92.41%	68.90%	94.67%	70.03%
dc:dl:db = 1:8:1	85.90%	92.66%	70.08%	94.67%	70.84%
dc:dl:db = 1:1:8	86.73%	92.91%	67.32%	**95.33%**	70.57%
dc:dl:db = 1:1:1	86.35%	92.15%	68.50%	**95.33%**	69.75%
dc:dl:db = 1:2:2	86.58%	92.15%	69.29%	**95.33%**	70.57%
dc:dl:db = 2:1:2	86.28%	91.90%	68.50%	94.67%	69.48%
dc:dl:db = 2:2:1	**86.81%**	**93.16%**	**71.26%**	94.00%	**71.93%**

Table 16: Overview of the five benchmark test datasets used for evaluation, including sample size, answer format, and type of task.

Dataset	Sample Size	Answer Format	Type of Task
GSM8K	1319	Number	Arithmetic reasoning
AddSub	395	Number	Arithmetic reasoning
AQuA-RAT	254	A/B/C/D/E	Arithmetic reasoning
Date Understanding	369	M/D/Y (date)	Semantic logical reasoning
Last Letter	150	String	Semantic logical reasoning

from the model's inability to maintain a coherent and anchored understanding of the question throughout the reasoning process.

A representative example of question-comprehension hallucination is presented in Table 18.

The Instruction Generator extracts the instruction associated with each intermediate reasoning step and compiles them into a sequence. During backward verification, the original question—"How many years will it take before he starts earning money on the lemon tree?"—is masked. The model is then prompted to infer the most plausible original question based solely on the extracted instructions.

As shown in Table 18, the model hypothesizes: "How many years will it take for Carlos to break even on the lemon tree?" A semantic comparison reveals a mismatch in intent. This reasoning chain is flagged as hallucinated due to the model's misinterpretation: while Carlos breaks even in year 12, when total profit equals the initial investment, he only begins to earn net profit in year 13, when cumulative profit exceeds the cost. The model's conflation of "start earning" with "break even" leads to an incorrect answer.

A.10 Commonsense Supplement in Condition Display Module

The focus of the Condition Display Module is to integrate and refine the conditional information provided by the original problem, collecting as much problem-relevant content as possible while filtering out irrelevant parts, so that the reasoning chain can rely on a more complete and accurate set of conditions. A key step in this process is the supplementation of commonsense—not as generic background facts, but as relational knowledge that connects decomposed conditions to each other and to the target question. By explicitly surfacing these hidden links, the module reduces the likelihood of

Table 17: Key codes of extraction functions for different reasoning tasks

Extracted Content	Key Code of Extraction Function
number	`match = re.search(r"####\s*([\d,]+(?:\.\d+)?)", ans_str)`
A/B/C/D/E	`match = re.search(r"####\s*([A-E])\b", ans_str)`
date information	`match = re.search(r"####\s*(.+)", ans_str)`
letter	`match = re.search(r'the␣answer␣is\s+"([^"]+)', ans_str, re.IGNORECASE)`

implicit fallacies during reasoning. Unlike previous commonsense augmentation approaches, our method neither calls any external interfaces nor supplements broad background knowledge that may introduce noise. Instead, it relies solely on the large model's own capacity to generate concise commonsense that strengthens the relational connections among conditions and between conditions and the problem statement.

First, the Condition Display Module supplements the necessary commonsense according to the relational structure of the problem. This involves identifying missing dependencies or associations among conditions and between conditions and the query, and then making them explicit within the structured condition list. With the advent of the latest large models such as DeepSeek R1 and Qwen 3, we believe these models are already capable of capturing most of the required relational commonsense; external APIs would not only consume more time and computation but also risk introducing irrelevant background information.

The supplementation of commonsense varies with the type of the problem. For example, when date calculation in February is involved, it can supply the relation between "leap year" and "number of days," clarifying how a background condition modifies the main query; if dice-probability calculation is required, it can link the property of dice faces to the probability sub-conditions. In addition, we observe that many problems involve counting items of certain categories, where distinguishing relevant from irrelevant categories is crucial. For instance, when counting fruits, the original conditional information may list the number of sandwiches as a distraction; supplementing the commonsense relation that "sandwiches are not fruits" explicitly links this condition back to the actual question, thereby preventing errors. Representative examples of such commonsense supplements are shown in Table 19.

Finally, in some cases, no additional commonsense is required because the conditions and their relations to the question are already explicit. In such cases, the large model is prompted to output "no additional commonsense needed," so as to avoid redundant information that might obscure the condition–question relationship.

A.11 Selection of k in Candidate Chain-of-Thought

Although a larger number k of candidate chains of thought makes the selection of the final answer more reliable, the choice of k must also take into account computational and time costs. Multi-dimensional verification filters candidate chains from multiple aspects—conditions, commonsense, and logic—often eliminating more candidates; therefore, it is necessary to increase k to ensure that a sufficient number of chains remain after screening, so that subsequent Majority Voting has enough samples for voting.

This method proposes *Discard-Weighted Voting*, which, compared with Majority Voting, does not reject an entire candidate chain because of a small error within it, thereby effectively alleviating the need to raise k under multi-dimensional verification and improving the efficiency of chain-of-thought reasoning.

We sampled 50 question–answer pairs from each of the five datasets and averaged the results to evaluate the effect of different values of k on the final answer accuracy; see Table 20 for details.

A.12 Why we score question semantic similarity by LLMs

In our *Reverse Instructional Problem Backtracking Verification*, we use a large language model (LLM) to score the semantic equivalence between the predicted question q' and the original question q, rather than relying on cosine similarity over sentence embeddings (e.g., SBERT). We choose LLM-based scoring because it is more robust to rephrasings, word order changes, and syntactic variations, and thus better at recognizing "same meaning, different phrasing."

As an illustration, original q: "How many U.S. states border Colorado?"; predicted q': "How many states are adjacent to Colorado in the United State?" The two differ in wording ("border" vs. "adjacent") and structure, but target the same meaning (the count of neighboring states). For such cases, LLM-based equivalence scores are typically higher and more reliable than static-embedding cosine similarity, which can be sensitive to lexical and positional shifts.

As also reflected in Table 1, backward verification for question-comprehension hallucinations trails other checks. This gap aligns with the volatility of LLM-produced semantic scores discussed above, which in turn makes the choice of a universal threshold τ uncertain. In this paper we report $\tau = 0.8$ based on validation data for our main setup, but we do not claim universality; as noted in our limitations, future work should examine varying τ and broaden evaluation across different LLMs.

A.13 Manually Annotated Verification Dataset

We construct two manually annotated datasets, each containing 250 questions paired with model-generated reasoning chains. The two datasets share exactly the same set of questions but differ in the hallucination types they evaluate. The first dataset includes reasoning chains that may exhibit any subset of the three hallucination categories—condition hallucination, logic hallucination, and question-comprehension hallucination (ranging from 0 to 3 types). The second dataset, in contrast, contains only question-comprehension hallucinations. Its purpose is to evaluate the computational cost and accuracy of different backward-verification methods.

Dataset for multi-dimensional hallucination analysis. To systematically evaluate MHCV's ability to identify all three types of hallucinations, we construct a 250-sample dataset with fine-grained human annotations. Specifically, we randomly sample 50 questions each from GSM8K, AddSub, AQuA-RAT, Last Letter, and Date Understanding. For every question, we generate ten candidate reasoning chains using DeepSeek V3, from which annotators select 50 fully correct reasoning chains and their corresponding 50 questions. We then manually annotate the ground-truth hallucination labels and ensure per-dataset balance by including 25 chains with entirely correct intermediate reasoning and 25 chains containing at least one hallucination type.

Each reasoning chain is annotated along four dimensions:

- **Condition Hallucination:** Whether referenced values, entities, condition indices, or condition contents appear in the original problem or correspond to valid commonsense; whether supplemented commonsense aligns with natural or physical laws. Any fabricated or incorrect reference is labeled as `false`.
- **Logic Hallucination:** Whether intermediate steps follow valid arithmetic or semantic logic. Violations such as incorrect numerical operations or ill-defined deductive jumps are labeled as `false`.
- **Question-Comprehension Hallucination:** With the final answer masked, whether the reasoning chain remains aligned with the original question. Deviations such as answering a different question or omitting critical aspects result in a `false` label.
- **Overall Correctness:** A holistic judgment integrating all three dimensions as well as the correctness of the final answer.

Dataset for backward-verification analysis. The second dataset uses the same 250 questions but is annotated solely for backward question-comprehension hallucination. For each question, half of the reasoning chains are labeled as entirely correct, while the other half contain explicit comprehension hallucinations. This dataset is used to evaluate the computational overhead and accuracy of our backward-verification method compared with baseline approaches.

Annotation protocol. We manually constructed two benchmark sets, each containing 250 samples. Five annotators with research backgrounds in NLP participated in the annotation process. Each sample was independently labeled by two annotators for the presence of three types of hallucinations: condition hallucination, logical hallucination, and question-comprehension hallucination. We report Cohen's κ scores of 0.85 (condition), 0.82 (logic), and 0.75 (question-comprehension), with an overall κ of 0.81. Disagreements were resolved by a third annotator to determine the final label. Due to the semantic diversity across different questions, ambiguity may arise in determining the precise semantic context during annotation, which contributes to the relatively lower agreement score for question-comprehension hallucination compared to the other two categories. Detailed annotation guidelines and procedures are provided above. Although the dataset size remains relatively limited, we plan to expand the benchmark in future work by involving additional annotators.

A.14 Analysis of Semantic-Scoring Instability in Backward Verification

In the backward-verification module, we abstract the reasoning chain into a sequence of structured instructions and compute a semantic similarity score $s(q, \hat{q})$ between the reconstructed question $\hat{q}$ and the original question q. This score is used to determine whether a question-comprehension hallucination has occurred. Unlike condition and logic verification—which rely on discrete, explicitly checkable constraints—semantic equivalence is an open-ended natural-language matching problem. It is inherently sensitive to variations in phrasing, paraphrasing, word order, redundancy, and stylistic differences. As a result, the distribution of semantic scores can vary considerably across datasets, tasks, and even samples within the same dataset. Semantically equivalent paraphrases may receive unexpectedly low scores, while partially mismatched reformulations may be assigned artificially high scores. Such instability can cause fixed-threshold decisions based on τ to behave inconsistently, reducing the robustness of backward verification.

Moreover, semantic similarity operates in a continuous "soft matching" space and lacks the discrete, structurally observable constraints that make condition and logic verification highly reliable. Consequently, both the false-positive and false-negative rates of semantic scoring are substantially higher than those of the other two detectors. In practice, this manifests in two characteristic failure modes: (i) *over-penalization*, where reasoning chains that correctly solve the original problem are erroneously rejected due to benign paraphrasing differences; and (ii) *under-penalization*, where chains that partially deviate from the intended question are mistakenly accepted. This instability further propagates into the final Discard-Weighted Voting process, motivating our choice to assign a relatively small weight to the backward-verification dimension. Our sensitivity analysis across different values of τ highlights this dependence on the score distribution.

To address these issues, we plan to improve the semantic-scoring mechanism in future work. One direction is to adopt an adaptive thresholding strategy, where τ is selected using a small development set tailored to the target task and dataset distribution. Another direction is to incorporate more structured signals—such as condition coverage, target-quantity alignment, and reasoning-path consistency—to complement the inherently noisy semantic score. These enhancements aim to reduce distributional instability while preserving the interpretability benefits of backward verification.

A.15 Weight Allocation for Condition, Logic, and Question-Comprehension Verification

In Appendix A.7, we experimented with different weight combinations for condition hallucination, logic hallucination, and question-comprehension hallucination, and evaluated their impact on the final-answer accuracy across all datasets. Empirically, the configuration $\omega_{\text{cond}} : \omega_{\text{logic}} : \omega_{\text{backward}} = 2 : 2 : 1$ achieved the strongest performance on the arithmetic and semantic–logical reasoning tasks considered in this paper. In this section, we provide a theoretical explanation for why this ratio performs best, and discuss how the weight assignment may change for future tasks.

To analyze the relative importance of the three hallucination detectors, we treat each detector V_t ($t \in \{1, 2, 3\}$, corresponding respectively to condition verification, logic verification, and question-comprehension verification) as a binary classifier for the latent correctness label $y \in \{0, 1\}$, where $y = 1$ denotes a reliable reasoning chain and $y = 0$ denotes an unreliable one. For detec-

tor V_t, we denote its binary decision function as $\phi_t(C) \in \{0, 1\}$, where C is the observed reasoning chain and $\phi_t(C) = 1$ means that the detector judges the chain as *passing* (i.e., reliable), while $\phi_t(C) = 0$ means the chain is judged as *failing*.

We define the false-positive rate (FPR) of detector V_t as

$$\alpha_t = \Pr\big(\phi_t(C) = 1 \mid y = 0\big),$$

which measures how often the detector incorrectly marks an *unreliable* reasoning chain ($y = 0$) as passing.

Similarly, the false-negative rate (FNR) is defined as

$$\beta_t = \Pr\big(\phi_t(C) = 0 \mid y = 1\big),$$

which measures how often the detector incorrectly rejects a *reliable* reasoning chain ($y = 1$).

We compute an overall error level as

$$\varepsilon_t = \tfrac{1}{2}(\alpha_t + \beta_t),$$

and define an "observability" indicator

$$\delta_t = 1 - \alpha_t - \beta_t,$$

which quantifies how well detector V_t separates reliable from unreliable reasoning chains. A higher δ_t indicates stronger discriminative ability and lower misclassification noise.

Condition verification and logic verification rely primarily on discrete, symbol-level constraints such as value referencing, entity matching, and arithmetic consistency. These checks are highly observable and tend to have low α_t and β_t, leading to larger δ_t and smaller ε_t. In contrast, question-comprehension verification must reconstruct the underlying question from a reasoning trace and evaluate semantic equivalence. This process is inherently noisier and more sensitive to paraphrasing, ambiguity, and information redundancy, resulting in significantly higher $\varepsilon_{\text{backward}}$ and lower δ_{backward}. Refer to appendix A.14 for cause analysis. Overall, the detectors typically satisfy

$$\delta_{\text{cond}} \approx \delta_{\text{logic}} > \delta_{\text{backward}},$$
$$\varepsilon_{\text{cond}} \approx \varepsilon_{\text{logic}} < \varepsilon_{\text{backward}}.$$

Under the standard assumption that detectors are approximately conditionally independent given the true label, the posterior log-odds

$$\log \frac{\Pr(y = 0 \mid \phi_1, \phi_2, \phi_3)}{\Pr(y = 1 \mid \phi_1, \phi_2, \phi_3)}$$

can be expressed as a linear function of the detector outputs:

$$\log \frac{\Pr(y = 0 \mid \phi_1, \phi_2, \phi_3)}{\Pr(y = 1 \mid \phi_1, \phi_2, \phi_3)} \approx \text{const} + \sum_{t=1}^{3} g_t\big(\phi_t(C)\big),$$

where each increment $g_t(\cdot)$ is monotonically related to the detector's error level ε_t. Detectors with lower ε_t contribute more strongly to the posterior decision boundary, while detectors with higher ε_t contribute less. The weighted discard score used in MHCV,

$$\text{Score}_a(i) = \sum_{t=1}^{3} \omega_t \cdot \phi_t(C_i),$$

can thus be viewed as a parametric linear approximation to the above log-odds. Because condition and logic verification exhibit significantly lower noise compared to question-comprehension verification, a natural ordering emerges:

$$\omega_{\text{cond}} \approx \omega_{\text{logic}} > \omega_{\text{backward}}.$$

The empirical choice of $2 : 2 : 1$ is a coarse but effective instantiation of this ordering. It reflects the fact that condition and logic verification provide similarly reliable evidence and should dominate the decision process, while question-comprehension verification—despite being essential for capturing semantic-level deviations—contributes noisier signals and therefore should be assigned a smaller weight. Our ablation results in Appendix A.7 confirm that this asymmetric allocation provides the strongest performance among the evaluated weight configurations and improves selection stability under multi-dimensional verification.

For future tasks, however, the optimal weight distribution may not remain fixed at $2 : 2 : 1$. In domains such as open-domain QA or multi-turn dialogue, question-comprehension errors may be more prevalent, and the semantic verification component can often be improved through task-specific training or adaptation, thereby reducing $\varepsilon_{\text{backward}}$ and increasing δ_{backward}. In such settings, weights may naturally shift toward more balanced configurations, such as $1 : 1 : 1$ or even $1 : 1 : 2$, depending on the relative reliability of each detector. From this perspective, the $2 : 2 : 1$ configuration used in this paper is best interpreted as a task-aligned instantiation rather than a universal hyperparameter.

A.16 Usage of Large Language Models

We used LLM solely to aid in polishing the writing of this paper after the main content had been completed. The LLM was not used for ideation, retrieval of related work, experimental design, analysis, or any other aspect of the research process. All conceptual, technical, and experimental contributions are original and were carried out independently.

A.17 Potential Risks

This work proposes PRISM to improve the reliability of chain-of-thought (CoT) reasoning through multi-dimensional verification. Despite its effectiveness, the framework inherits limitations from its reliance on large language models for both reasoning and verification, which may allow systematic biases or misinterpretations to persist across stages.

The proposed approach introduces additional computational overhead by generating multiple reasoning chains and performing both forward and backward verification. This increased cost may limit applicability in resource-constrained settings and raises concerns about efficiency when the method is applied at scale.

While PRISM improves reasoning accuracy on established benchmarks, it does not guarantee correctness in high-stakes real-world applications such as medical, legal, or financial decision-making. Moreover, the evaluation is conducted on standard datasets with limited manual annotations, and the reported gains may not fully generalize to more complex or adversarial scenarios.

A.18 Reasoning and Verification Prompts Used in Experiments

This appendix presents the complete prompts used for the GSM8K dataset in our experiments. The prompts for other arithmetic, semantic logical reasoning, and SQL application datasets are adapted from this base structure according to task type, implementing the CPIR and MHCV chain-of-thought reasoning and verification framework proposed in this paper.

The model first generates candidate reasoning chains in the CPIR stage, explicitly referencing conditions and incorporating relevant common knowledge when necessary. Then, in the MHCV stage, each reasoning chain is verified and filtered across three dimensions: condition consistency, logical consistency, and question comprehension consistency. Finally, a discard-weighted voting strategy is applied to determine the final answer.

Specifically, the reasoning-generation prompt (shown in Table 21) guides the model to decompose the original question into structured labeled conditions, introduce relevant common knowledge when information is missing, and produce a step-by-step reasoning process via progressive instructions.

The forward condition-consistency verification prompt (Table 22) checks whether any numerical values or factual references in the reasoning are unsupported by the original conditions or prior conclusions, and whether any injected common knowledge conforms to natural or social rules.

The forward logical-consistency verification prompt (Table 23) detects arithmetic errors or semantic logical inconsistencies within the reasoning chain.

The backward question-consistency verification prompt (Table 24) reconstructs the original question from the processed instruction list to detect any misunderstanding or semantic deviation in the reasoning process.

To ensure reproducibility and fairness, all experiments on the GSM8K dataset adopt a unified prompt structure and identical generation parameter settings, with only the underlying large language model replaced. Each table includes a one-shot example illustrating the corresponding reasoning or verification procedure.

A.19 Comparison with Alternative Semantic Equivalence Verifiers

To further contextualize the proposed reverse instructional backtracking verifier, we compare it against several strong and directly relevant semantic equivalence baselines. These alternatives evaluate whether the reconstructed question is semantically equivalent to the original question without relying on the instructional trajectory.

We evaluate four approaches: (1) **Calibrated sentence embeddings**, using `all-mpnet-base-v2` with logistic regression calibration and threshold selection; (2) **Lightweight cross-encoder**, based on `cross-encoder/nli-MiniLM2-L6-H768`; (3) **Bidirectional NLI**, based on `roberta-large-mnli`, where entailment probabilities are computed in both directions and averaged; (4) **Our Method**, prompting a large language model (`gpt-3.5-turbo`) to output a semantic equivalence score between two questions.

All methods are evaluated on the annotated

backward-verification subset derived from the 250-sample benchmark. We report Accuracy, F1-score, and AUC. In this evaluation setting, the model is not required to regenerate a predicted question from the instructional trajectory. Instead, the predicted question is directly extracted and compared with the original question. This setup is specifically designed to assess the model's ability to perform semantic equivalence comparison at the question level, rather than its capacity to reconstruct questions through instruction-based reasoning. Consequently, the accuracy in this setting is expected to be higher than that of the full reverse backtracking verification, since it isolates semantic comparison without introducing reconstruction difficulty.

As shown in Table 25, LLM-based semantic equivalence scoring using `gpt-3.5-turbo` achieves the highest overall performance, with an accuracy of 96.67% and an F1-score of 0.9600. Although the bi-directional NLI model attains a slightly higher AUC (0.9955), its lower accuracy and F1-score suggest reduced robustness under threshold-based decision settings.

Embedding-based calibration performs poorly in terms of accuracy, indicating that cosine similarity alone struggles to capture nuanced semantic equivalence in reconstructed questions. The lightweight cross-encoder also underperforms compared to the LLM-based approach, suggesting that smaller learned classifiers may lack sufficient capacity for fine-grained question-level semantic alignment.

Table 18: Illustrative Example of Question-Comprehensive Hallucination

Question	Carlos is planting a lemon tree. The tree will cost \$90 to plant. Each year it will grow 7 lemons, which he can sell for \$1.5 each. It costs \$3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?
Ground Truth	He makes \$10.5 selling lemons each year because $7 \times 1.5 = 10.5$. He earns \$7.5 each year from the lemon tree because $10.5 - 3 = 7.5$. It will take 12 years to pay off the tree because $90/7.5 = 12$. He will make money in year 13 because $12 + 1 = 13$. #### 13
Model Answer	#1. The cost to plant the lemon tree is \$90. #2. The lemon tree grows 7 lemons each year, which can be sold for \$1.5 each. #3. It costs \$3 a year to water and feed the tree. #4. How many years will it take before Carlos starts earning money on the lemon tree? #5. Carlos will start earning money when revenue exceeds total cost. #6. (by #2) Step 1: Calculate the annual revenue from selling lemons. (Instruction 1) Number of lemons per year = 7 Price per lemon = \$1.5 Annual revenue = $7 \times 1.5 = 10.5$ #7. (by #3) Step 2: Calculate the annual recurring cost of watering and feeding. (Instruction 2) Annual maintenance cost = \$3 #8. (by #1, #6, #7, #5) Step 3: Set up the equation to find the break-even year. (Instruction 3) Let x be the number of years. Total revenue over x years = \$10.5x Total cost = Initial planting cost (\$90) + Annual maintenance cost (\$3x) Break-even point: \$10.5x = \$90 + \$3x \$10.5x - \$3x = \$90 \$7.5x = \$90 x = \$90 / \$7.5 = 12 #9. (by #4, #8) So the answer to this question is after 12 years Carlos will begin to earn profit. (Hallucination) #### **12 (The correct answer is 13)**

Table 19: Examples of different types of example questions and their corresponding commonsense supplements (Part 1).

Common-sense Type	**Example Question**	**Commonsense Supplement**
Time & Calendar	In 2023, a streaming pass covers every day of February plus the first 3 days of March. In 2024, the same pass covers every day of February plus the last 4 days of January. A family buys one pass each year and uses it on every covered day. Across both years, how many total days are covered by the passes?	February has 28 days in common years and 29 days in leap years.
Item Category	A deli packs meal kits: each kit has 4 apples, 3 bananas, 2 oranges, 5 cupcakes, and 1 bottle of juice. Nora buys 3 kits, then during the week she eats 7 pieces of fruit, gives 5 pieces of fruit to a neighbor, and throws away 2 cupcakes that got squished. Counting only fruit items as single pieces and ignoring non-fruit, how many pieces of fruit does Nora have left?	Apples, bananas, and oranges are fruit; cupcakes and bottled juice are not counted as fruit pieces.
Units & Measurements	A craft shop sells beads in mixed containers: 2 boxes of one dozen beads each, 3 tubes of 25 beads, and a bulk bag holding half a gross. During a workshop, 19 beads roll away and are lost, and a student later finds 6 of the lost beads and returns them. How many beads remain after the workshop?	One dozen equals 12; a gross equals 144 (so half a gross is 72).
Currency & Value	Kaylee has 2 rolls of quarters (40 quarters per roll), 1 roll of nickels (40 nickels per roll), and loose coins: 7 quarters, 6 dimes, 4 nickels, and 12 pennies. She pays $23.87 for supplies using only these coins, handing over the largest values first until she reaches or exceeds the price, and receives no change. How many cents does she have left afterward?	Quarter = 25¢, dime = 10¢, nickel = 5¢, penny = 1¢.

Table 19: Examples of different types of example questions and their corresponding commonsense supplements (Part 2).

Common-sense Type	**Example Question**	**Commonsense Supplement**
Days & Scheduling	A weekend workshop runs for 6 consecutive weeks and meets on both Saturday and Sunday each week. The organizer cancels all meetings on the 3rd weekend, then adds make-up meetings on the next two Mondays. Later, a surprise guest gives one extra session on a Sunday during the final week. Over the whole period, how many meetings are held?	Weekend days are Saturday and Sunday (2 per weekend).
Mathematical Properties	A school has lockers numbered 1 through 120. Students may choose a locker only if its number is prime and not a multiple of 3 or 5 (so numbers like 3 or 5 are excluded). The student council also removes any choice above 100 to keep assignments near classrooms. How many locker numbers remain available?	A prime number has exactly two positive divisors (1 and itself), and 1 is not prime.
Relationships & Classification	Rina has two siblings: an older brother with 3 children (including one set of twins counted as two children) and a younger sister with 2 children. Rina's spouse's sister has 1 child, and Rina's cousin has 2 children. For a family party, Rina is bringing gifts only for her own nieces and nephews. How many gifts does she need?	Your siblings' children are your nieces/nephews; cousins' children and your in-laws' children are not your nieces/nephews.

Table 20: Effect of different values of k on the overall final answer accuracy across the five datasets

	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 10$
Accuracy	73.76%	78.35%	79.72%	81.05%	81.77%

Table 21: GSM8K Reasoning Generation Prompt (CPIR Reasoning Stage)

Module	**Prompt Content**
Question	Mary spends $3 a day on lunch in the cafeteria. She bought a $5 monthly membership card that gives her a 20% discount. How much did she spend on lunch in December?
Answer (Condition Extraction)	First, let's extract the necessary information and rewrite the question using labels.
	#1. Mary spends $3 a day on lunch in the cafeteria.
	#2. She bought a $5 membership card.
	#3. The membership card gives her a 20% discount.
	#4. How much did she spend on lunch in December?
Common Knowledge Injection	Next, we add the necessary knowledge from the question if needed.
	#5. December has 31 days.
Step-by-Step Reasoning	Finally, let us solve the problem step by step with reference to the question and reasoning process:
	#6. (by #1, #3) Step 1: Calculate the discounted daily cost.
	Normal daily cost: $3, Discount: 20%.
	Discounted daily cost = $3 × (1 - 0.20) = $3 × 0.80 = $2.40.
	#7. (by #5, #6) Step 2: Calculate the total cost for December lunches.
	December has 31 days, daily discounted cost: $2.40.
	Total discounted lunch cost = 31 × $2.40 = $74.40.
	#8. (by #2) Step 3: Include the cost of the membership card.
	Membership card cost: $5.
	#9. (by #7, #8) Step 4: Determine Mary's total spending.
	Total lunch cost (discounted): $74.40.
	Membership card cost: $5.
	Total spent = $74.40 + $5 = $79.40.
Final Answer	#10. (by #4, #9) The original question is #4: How much did she spend on lunch in December?
	We do not miss information in the rewritten labels.
	Therefore, the answer is: In December, Mary spent $79.40.

Table 22: GSM8K Forward Condition-Consistency Verification Prompt

Module	**Prompt Content**
Given Details	Here is a set of details:
	#1. Jenny pays $2 for her daily bus ticket.
	#2. She has a membership card that reduces the daily fare by 20%.
	#3. The membership card costs $5.
	#4. April has 30 days.
Intermediate Reasoning Steps	Based on these details, here is a reasoning process:
	Step #5 (by #1, #4): Jenny's total bus fare for April is 30 × $2 = $60.
	Step #6 (by #2, #5): After applying a 20% discount, her fare is $60 × (1 - 0.20) = $48.
	Step #7 (by #3, #6): Adding the membership card cost of $5, her final payment is $48 + $5 = $53.
Verification Instruction	Double-check the numeric references and verify the correctness of any common knowledge used.
	End with "yes" or "no" to indicate whether the condition check passes.

Table 23: GSM8K Forward Logical-Consistency Verification Prompt

Module	**Prompt Content**
Given Details	Here is a set of details:
	#1. Jenny pays $2 for her daily bus ticket.
	#2. She has a membership card that reduces the daily fare by 20%.
	#3. The membership card costs $5.
	#4. April has 30 days.
Intermediate Reasoning Steps	Based on these details, here is a reasoning process:
	Step #5 (by #1, #4): Jenny's total bus fare for April is 30 × $2 = $60.
	Step #6 (by #2, #5): After applying a 20% discount, her fare is $60 × (1 - 0.20) = $45.
	Step #7 (by #3, #6): Adding the membership card cost of $5, her final payment is $45 + $5 = $50.
Verification Instruction	Verify whether the reasoning process is internally consistent.
	End with "yes" or "no".

Table 24: GSM8K Backward Question-Consistency Verification Prompt

Module	**Prompt Content**
Intermediate Reasoning Steps	Here is a set of reasoning steps:
	Step 1: Calculate the discounted daily cost.
	Step 2: Calculate the total cost for December lunches.
	Step 3: Include the cost of the membership card.
	Step 4: Determine Mary's total spending.
Original Question	How much did she spend on lunch in December?
Verification Instruction	Reconstruct the original question based only on the reasoning steps. Do not output it.
	End with "yes" or "no" to indicate whether semantic consistency holds.

Table 25: Comparison of semantic equivalence verification methods on backward-verification data

Method	**Accuracy**	**F1**	**AUC**
Calibrated Embedding (MPNet)	0.5000	0.6341	0.7602
Lightweight Cross-Encoder (MiniLM)	0.6333	0.6207	0.8100
Bi-directional NLI (RoBERTa-MNLI)	0.8667	0.8182	**0.9955**
Our Method (`gpt-3.5-turbo`)	**0.9667**	**0.9600**	0.9932